VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Stereo korespondence

Stereomatch

2012

Roman Meca

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Roman Meca**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Stereo korespondence
Stereomatch

Zásady pro vypracování:

Stereo korespondence je jedním ze základních problémů v oblasti rekonstrukce 3D scény. Jedná se o těžký problém, který doposud není uspokojivě vyřešen. Nalezení korespondence se často formuluje jako globální optimalizační problém, která zahrnuje všechna možná geometrická a fotometrická omezení jako je například zákryt nebo průhled.

1. Seznamte se s algoritmy stereo korespondence.
2. V práci shrňte stávající přístupy a metody.
3. Implementujte vybraný algoritmus pro stereo korespondenci využívající optimalizaci pomocí Belief propagation (BP) nebo Graph Cuts (GC)
4. Optimalizujte implementovaný algoritmus (MMX, SSE, vlákna nebo CUDA).
5. Navrhněte metodu porovnávající jejich úspěšnost pro zadané dvojice snímků.

Seznam doporučené odborné literatury:

Boguslaw Cyganek, An Introduction to 3D Computer Vision Techniques and Algorithms, Wiley; 1 edition, 2009, ISBN 978-0470017043
E. R. Davies, Machine Vision, Third Edition: Theory, Algorithms, Practicalities, Morgan Kaufmann; 3 edition, 2005, ISBN 978-0122060939
http://vision.middlebury.edu/stereo/

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Michal Krumnikl**

Datum zadání: 18.11.2011
Datum odevzdání: 04.05.2012

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 4. května 2012 .....

Rád bych na tomto místě poděkoval všem, kteří mi s prací pomohli. Protože bez nich by tato práce vznikala pouze velmi obtížně. Hlavní díky patří vedoucímu práce Ing. Michalu Krumniklovi za jeho trpělivost, cenné rady a postřehy. V neposlední řadě také mé rodině. Ta mě v mém snažení všemožně podporovala a poskytla potřebné zázemí a klid potřebný k tvorbě této práce.

Abstrakt

Tato diplomová práce popisuje řešení problému stereo korespondence. Ten lze zjednodušit na problém nalezení korespondujících bodů v sadě záznamů 3D scény. Hledaným výsledkem je relativní vzdálenost těchto dvou bodů ta se nazývá disparita. S její pomoci lze snadno spočítat polohu pozorovaného bodu vůči pozorovateli.

Práce stručně shrnuje historii vývoje a výzkumu v oblasti rekonstrukce 3D scény a úzce souvisejících odvětvích. Hlavně však popisuje základní principy a způsoby dělení různých druhů algoritmů vyvinutých k tomuto účelu.

Zvláště je pak rozebrána teorie nutná k porozumění a pochopení principů globálního algoritmu Belief propagation a možné způsoby optimalizace tohoto algoritmu. Ta je realizována hlavně za pomoci procesní optimalizace a pomocí vláken. Ovšem jsou zmíněny také jiné možnosti optimalizace jako SSE, CUDA nebo novější OpenCL.

Klíčová slova: Korespondence v obrazech, Stereoprojekce, Disparita, Disparitní mapa, Epipolární geometrie, Belief propagation, SSE, Vlákna, CUDA, OpenCL

Abstract

This thesis describes how to solve the problem of stereo correspondence. This can be simplified for finding matching points in the recordset 3D scene. Searched result is the relative distance of these two points is known as the disparity. If you have enough data then with this value you can easily calculate the position of the observed points towards the observer.

Thesis briefly summarizes the history of the development and research in the field of 3D scene reconstruction and closely related sectors. Mainly, however, describes the basic principles and methods of separation of different types of algorithms developed for this purpose.

In particular, it analyzes the theory necessary for understanding of global Belief Propagation algorithm and possible ways to optimize this algorithm. Optimalization is implemented using threads and SSE CPU instructions. But others ways to optimize this algoritm are also mentioned as CUDA or OpenCL.

Keywords: Correspondence in paintings, Stereoprojekce, Disparity, Disparity map, Epipolar geometry, Belief Propagation, SSE, Threads, CUDA, OpenCL

Seznam použitých zkratek a symbolů

BP	–	Belief Propagation
SAD	–	Sum of Absolute Difrences
SSD	–	Sum of Square Difrences
SSD-N	–	Sum of Normalized Square Difrences
ZSAD	–	Zero mean Sum of Absolute Difrences
ZSSD	–	Zero mean Sum of Square Difrences
ZSAD-N	–	Zero mean Normalized Sum of Absolute Difrences
ZSSD-N	–	Zero mean Normalized Sum of Square Difrences
CV	–	Covariance - Variance
SCP	–	Sum of Cross Products
SCP-N	–	Normalized Sum of Cross Rroducts
POSIX	–	Portable Operating System Interface
PC	–	Personal Computer
VŠB	–	Vysoká Škola Báňská
CNT	–	Czech National Team
CPU	–	Central Procesing Unit
GPU	–	Graphical Procesing Unit
FLOPS	–	FLoating-point OPerations per Second
CUDA	–	Compute Unified Device Architecture
OpenCL	–	Open Computing Language
SIMD	–	Single Instruction, Multiple Data
SSE	–	Streaming SIMD Extensions
MMX	–	Matrix Math eXtension
CCD	–	Charge-coupled device
CMOS	–	Complementary Metal–Oxide–Semiconductor
DARPA	–	Defense Advanced Research Projects Agency
DPI	–	Dots per inch
API	–	Application Programming Interface
Pan-STARRS	–	Panoramic Survey Telescope & Rapid Response System

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Algoritmy stereometrické korespondence se v poslední době dostaly na výsluní pozornosti. Jednak díky nárůstu výkonu výpočetní techniky což nám umožňuje dosahovat výsledků z lidského hlediska v reálném čase. Druhou stránkou je jejich nezměrné využití jak v oblastech zábavy. Zde stojí za zmínku zařízení Kinect od společnosti Microsoft nebo Move od Sony. Tak i využití ve zdravotnictví a jiných odvětvích jako je robotika. Zde všude přináší možnost prostorového vidění přístroje nové možnosti. Problém stereo korespondence vychází z imitace lidského zraku a to především procesů, které probíhají v mozku.

Hardware dnešních kamer dnes již vcelku obstojně zvládá konkurovat schopnostem lidského oka a v některých oblastech ho dokonce dalece předčí. A to ať už se jedná o velikost vnímaného vlnového spektra, rozlišení, zorný úhel nebo možnost přiblížení vzdálených objektů. V případě rozlišení myslíme celkový počet bodů. Ne počet bodů na palec při porovnání hodnot dpi lidské oko pořád jasně vede a v blízké době se na tom nebude nic měnit vzhledem k technologickým omezením.

Momentálně je největší slabinou celého procesu právě samotný algoritmus řešení problému rekonstrukce 3D scény. Vzhledem k tomu že i lidský mozek se dá velmi snadno ošálit pomocí různých optických klamů, viz příloha A. Tyto klamy však většinou dokážou zmást pouze lidi nikoli stroj. Důvod je prostý, jednak pracují pouze s 2D imitací 3D prostoru. Následně jsou také využity zažité grafické souvislosti, které vidíme v každodenním životě. V počítačovém zpracování se většinou těchto souvislostí nevyužívá, tím sice dochází k jisté redukci funkčnosti. Z druhé strany odpadají mnohé vady lidského zraku, který z velké části funguje na zkušenosti. Detailněji se funkcí lidského zraku zabývá Ing. Michal Krumnikl ve své bakalářské práci [2].

Tudíž dnes již není překážkou ve strojovém napodobení lidského vidění a vnímaní prostoru hardware ale spíše software. Problém vzniká při pokusu reprodukovat a popsat procesy probíhajícími v mozku. Přesný popis procesů probíhajících v mozku při zpracování obrazové informace je předmětem mnoha výzkumů, avšak stále ještě nebyl dokonale popsán. Obecně se dá říci, že mnoha tajemství lidského mozku mezi nimi i způsoby jakým přesně zpracovává obrazovou informaci, zůstávají dosud neobjevena.

V procesu imitace lidského zraku dnešní věda však výrazně pokročila a zde právě přichází na řadu různé algoritmy stereo korespondence. Tyto algoritmy se již poměrně dlouhou dobu vyvíjí a prošly určitým vývojem. V dnešní době se algoritmy dělí hlavně na lokální a globální. Nás v této práci bude především zajímat algoritmus zvaný Belief Propagation spadající do kategorie globálních algoritmů. Nicméně zběžně si přiblížíme i některé jiné algoritmy zde stoji za zmínku publikace „An introduction to 3D komputer vision techniques and algorithms“ autorů Bogusława Cyganka a J. Paula Sieberta [1]. Tato kniha shrnuje všechny základní techniky využívané algoritmy stereo korespondentce, počítačového vidění a zpracovaní obrazu.

Nejlepší současné algoritmy ovšem nejsou kompletně publikovány vzhledem k jejich velkému komerčnímu potenciálu nebo v případě armády z důvodu dodržení utajení. Nicméně efektivních algoritmů je publikována celá řada a mají řadu možných využití a

za účelem měření správnosti a efektivity těchto algoritmu byly publikovaný Middlebury datasets [29]. Což jsou volně dostupné sady testovacích dat pro tyto algoritmy. Také je poskytnuta možnost zapsat výsledky vlastního algoritmu a získat srovnání s ostatními již zapsanými výsledky.

Tato práce se skládá z několika částí. První je kapitola 2 zabývající se historií a vývojem vědeckých poznatků a sumarizuje dalekou cestu, kterou věda musela urazit, než se dostala do tak pokročilého bodu jako dnes. Za touto krátkou kapitolou následuje popis geometrických vztahů nutných k porozumění našeho problému v kapitole 3. Na základě těchto vztahů je následně v kapitole 4 definován problém stereo korespondence. Pak již následuje přehled v současnosti dobře ověřených a používaných algoritmů ty shrnuje kapitola 5. Způsoby hodnocení a srovnání výsledku dosažených stero korespondenčními algoritmy jsou představeny v kapitole 6 Po těchto kapitolách již následují kapitoly zbývající se teorii algoritmu Belief Propagation kapitola 7, optimalizaci tohoto algoritmu kapitola 8. Předposlední část této práce kapitola 9 se zbývá naší implementací zvoleného stereo korespondenčního algoritmu Belief propagation. Závěrem práce v kapitole 10 shrneme dosažené výsledky a nastíníme možnou budoucnost vývoje v této oblasti.

2 Historie

Nejprve se pojďme podívat do historie. Co vlastně předcházelo vypracování této práce a jak postupně lidstvo objevovalo poznatky využité v této práci. V minulosti se často stávaly případy paralelních objevů. Tudíž že dva lidé nezávisle na sobě objevili a popsali tu samou věc. To dnes v době internetu již není problém vzhledem k rychlosti šíření informací internetem. I tak bychom měli studiu historie věnovat alespoň chvilku.

2.1 Starověk a středověk

Již od počátku lidské civilizace se člověk zajímal o způsob, jakým vnímá své okolí. Vědomosti o způsobu vidění což je nejdůležitější lidský smysl se vyvíjely nejprve v oboru medicíny. A to hlavně z praktického důvodu nutnosti ošetřit zranění nebo jako pokusy o vyléčení vad a nemocí zraku.

Tyto poznatky vznikaly dlouhou dobu a první zmínky jsou staré jako písmo samo. V tomto ohledu bychom neměli zapomenout zmínit starověké civilizace, jako byla Mezopotámie a Egypt následovaný antickým Řeckem a římskou civilizací. Po tomto počátečním období rozkvětu přichází v Evropě období nadvlády církve a exodu učenců na východ převážně do arabského světa. Tam nebyli pronásledováni a mohli v klidu tvořit a publikovat své práce.

Následně až v Evropě 13. až 17. století dochází k dalšímu rozvoji. Zde jmenujme učence a vynálezce jako jsou Leonardo da Vinci, Giovanni Battista della Porta, Benedetto Castelli a mnozí další. Tuto éru zakončil Johaness Kepler který pokládá základy moderní optiky, když roku 1611 vydal svůj spis Dioptrice.

2.2 Moderní věda 18. a 19. století

Moderní věda 18. a 19. století pokračovala o poznání rychleji a to jak z důvodu vynálezu mikroskopu. Tento vynález umožnil detailnější poznání funkce lidského oka. Zde není jasné, kdo mikroskop objevil první. Nicméně první zmínky mluví o holandském inženýrovi Corneliovi Drebbelim. Následně pak Sir Isaac Newton jako první správně popsal cestu obrazové informace skrz oko do mozku, toto své zjištění publikoval v díle Optika v letech 1704 – 1730.

Pro nás je důležitý článek Charlese Wheatstona publikovaný roku 1838 v časopise Royal Society. V něm dokazuje nezpochybnitelnou spojitost mezi disparitou a vnímanou prostorovou hloubkou. Toto zjištění je základem všech stereoskopických algoritmů.

Dokonce se objevil první koncept počítače. Ten je možno připsat Charlesi Babbagemu a první zmínka pochází z roku 1822. V této době taky probíhaly první pokusy s elektřinou. Zde jmenujme vědce, které dobře známe ze soustavy SI a to Alessandro Volta, Andreho Marie Ampera a George Simonse Ohma.

2.3 Současná věda

Všechny poznatky o elektřině byly využity ve 30. letech 20. století při sestrojení prvních číslicových počítačů. Ty by již mohly simulovat procesy probíhající v mozku nebo jinak hledat hodnotu disparity. Avšak jejich výkon nedosahoval potřebné výše. Zatímco rozměry těchto počítačů byly enormní. Z těchto důvodů nikdo ani nepomyslel na využití těchto strojů ke zpracování tak komplikovaného problému jako je zpracováni obrazové informace a hledání disparity. Využívala je převážně armáda k výpočtu balistických drah a dešifrování zpráv nepřítele. Většina dnešních kalkulaček tyto první počítače dalece předčí.

S vývojem elektroniky přicházely jak výkonnější tak i menší počítače. Tento trend umožnil mimo jiné rozvoj způsobu zobrazení a zachycení grafické informace na počítači. Následný vývoj byl již poměrně rychlý. Vedle vynálezu integrovaného obvodu ten sestrojil Jack St. Clair Kilby roku 1958 a vynálezu technologie CMOS Frankem Wanlassem v roce 1963 nebo následným objevením technologie CCD pány Willardem Boylem a Georgem E. Smithem roku 1969. Za tento poslední objev dostali jeho autoři roku 2009 Nobelovu cenu. Objevy všech těchto technologií vyvrcholily sestrojením prvního plně digitálního fotoaparátu a jeho uvedení na trh to proběhlo v roce 1988 pod názvem Fuji DS-1P.

V 70. a počátkem 80. let 20. století dochází k prvnímu vývoji počítačového zpracování obrazu a to pod záštitou organizace DARPA první prací shrnující tyto poznatky je dílo pánů Barnarda a Fischlera [4] publikované v roce 1981. Následně během 80. let probíhal v komunitě zbývající se počítačovým zrakem vývoj za účelem zdokonalení tehdy známých algoritmů. K tomuto účelu musely byt také vymyšleny a publikovány kriteria měření výkonu jednotlivých algoritmů. Případné zájemce lze odkázat na publikaci autorů Dhonda a Aggarwala, [5] v níž souhrnně publikovali tehdejší poznatky.

Počátkem 90. let již byly základní otázky vyřešeny. Tento stav dokládá publikace pánů Kanadeho a M. Okutomiho [10] zabývající se lokálními algoritmy s adaptivním okénkem. Přesto výzkum této oblasti stále pokračoval jen ne tak intenzivně jako na počátku. Došlo k rozmělnění komunity a většina výzkumníku obrátila svou pozornost k řešení detailnějších problému jako je problém překrytí, průhlednost a realtime zpracování. Podrobnější obrázek o tehdejším postupu bádání si lze udělat v nepublikované práci autora Koschana [6].

Podstatný pokrok byl učiněn v posledním desetiletí. Byly přijaty nové trendy v informatice čtenáři v tomto ohledu mohu jen doporučit knihy autorů Hartleyho a Zissermana [7] a Faugerase a Luonga [8]. Tyto dvě publikace poskytují rozsáhle informace o geometrických aspektech multipohledového sterea. Také nesmíme zapomenou na knihu pánů Scharsteina a Szeliskiho [9], která dopodrobna popisuje různé moderní přístupy k řešení problému stereo korespondence a následné optimalizace navržených algoritmů.

3 Základy epipolární geometrie

Jak už bylo zmíněno jedna se o vtahy popisující geometrické vlastnosti pozorovaného prostoru a soustavy pomocí které získáváme data. Dále pak popisuje vzájemné vztahy prostoru a optické soustavy, kterou tento prostor pozorujeme. Máme li dostatek informací lze dopočítat vzdálenost objektu nacházejícího se v pozorovaném prostoru od pozorovatele respektive optické soustavy. Přesnost tohoto výpočtu velmi závisí na parametrech soustavy. Obrázky využité k popisu jsou vzhledem k jejich počtu a velikosti umístěny v příloze B.

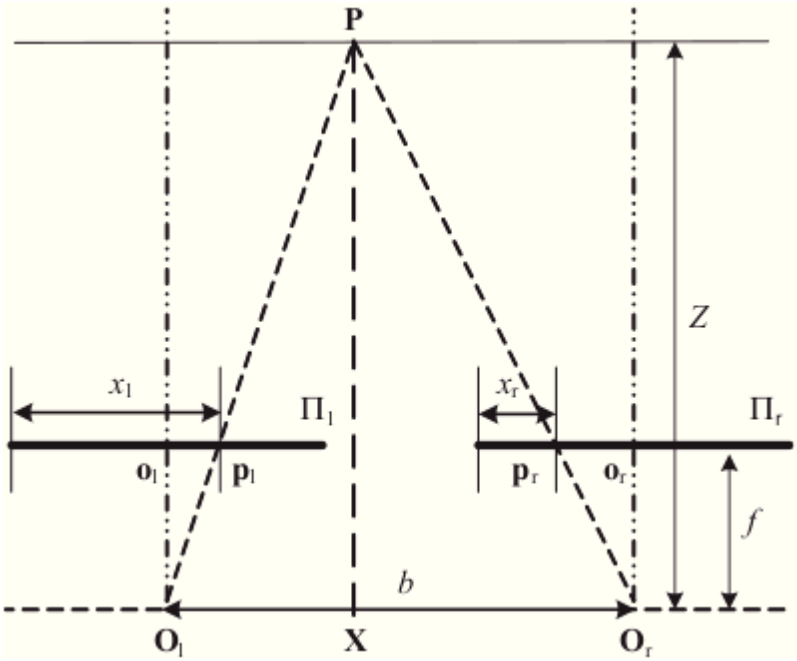
Nejlepší popisnou hodnotu má vždy příklad a proto si za něj zvolíme takový menší experiment. Ten si může každý snadno provést doma. Budeme potřebovat pouze pravítko, fix libovolné barvy a okno. Okno v našem experimentu bude reprezentovat 2D záznamy scény jeden pro levé oko a jedem pro pravé. Následně zvolíme pozorovaný objekt. Ten by měl být pokud možno statický vzhledem k časovému posunu mezi záznamem jednotlivých referenčních bodů. Zavřeme jedno oko a poznačíme si na skle fixem referenční bod nejlépe nějakou z hran objektu. Poté provedeme ten samý úkon s dosud zavřeným okem. Výsledkem by měly být dva různé body tvořící vodorovnou přímku.

Vodorovná je z důvodu uspořádání lidské optické soustavy. Oči jsou od sebe rozmístěny také na vodorovné přímce. Kdybychom si lehli a pozorovali prostor z pozice na boku. Byla by tato přímka vertikální a objekty by se pohybovaly nahoru a dolů. Výsledek prostorového vnímaní je ovšem stejný. V praxi se však většinou používá vodorovné uspořádání soustavy. Ovšem lze využít i mnohem komplikovanější optické soustavy nabízí se třeba uspořádání tří kamer do trojúhelníku apod.. Těmito případy se ovšem zabývat nebudeme.

Velikost přímky budeme nazývat velikostí disparity. Při delším experimentováni si všimneme dalších závislostí. Nejvýraznější je rozdílné chování vzdálených a blízkých objektu. Blízké objekty mají velkou disparitu. Zatímco v případě vzdálenějších objektů se disparita zmenšuje a pro dostatečně vzdálené objekty bude dokonce nulová. Tento jev si vysvětlíme později v této kapitole.

Naše vzorová Optická sousta je zobrazena na obrázku 1 a to z horního pohledu. Ten je nejnázornější a dají se na něm dobře popsat všechny důležité veličiny a vztahy. Opět při vertikálním uspořádaní soustavy by byl vhodnější pohled z boku. Pojďme si tedy nyní popsat, jak dojdeme k onomu kýženému výsledku což je vzdálenost pozorovaného objektu od pozorovatele.

Jak je vidět na obrázku 1 stereoskopická optická soustava se skládá ze dvou kamer v případě člověka očí. Na obrázku jsou označeny jako O_l a O_r důležitá pro nás bude také jejích vzdálenost označená jako b. Každá z těchto kamer má také optickou osu ty jsou značeny o_l a o_r. Ohnisková vzdálenost kamer je reprezentována úsečkou f. Pozorovaným objektem je pak bod P. Záznamy pozorované scény, ve kterých hledáme shodu, značíme jako množiny Π_l a Π_r. V nich je také vyznačeno zobrazení pozorovaného objektu jako p_l a p_r a vzdálenost těchto bodů od okraje množin x_l a x_r. Absolutní rozdíl těchto dvou vzdáleností se pak nazývá disparitní vzdálenost. Ve skutečnosti nás ovšem zajímá velikost přímky označené jako Z. Což je vzdálenost pozorovaného objektu od ohniska soustavy.



Obrázek 1: Optická soustava [1]

Když se nad obrázkem trošku zamyslíme, všimneme šesti pravoúhlých trojúhelníku a to dvou menších a čtyřech větších. Větší trojúhelníky jsou vždy ve dvojici zrcadleny podle středu osy. Ty to dvě osy tvoří vždy bod P a následně pak bod O_l nebo O_p pro každou osu jeden ze jmenovaných. Následně z tohoto faktu vyplývá jejich shoda. Proto budeme každý z těchto dvou párů dále uvažovat vždy jako jeden trojúhelník. Touto úpravou dostáváme dva menší trojúhelníky a dva větší. Menší trojúhelníky jsou tvořeny těmito vrcholy. Ohniskem kamery, průsečíkem optické osy a množiny reprezentující záznam scény pro danou kameru. Třetím bodem pak je zobrazeni pozorovaného objektu v záznamu scény.

Větší jsou pak tvořeny bodem X, bodem P a jedním z bodů O_l nebo O_r. Vzhledem k již zmíněnému zrcadlení navíc můžeme prohlásit velké a malé trojúhelníky za podobné a to hlavně podle věty o podobnosti trojúhelníku uu. Všechny trojúhelníky jsou pravoúhle což je první shodný úhel. Druhý utváří optická osa spolu s osou zrcadlení. Tu jsme využili k prohlášení o shodě a spárování velkých trojúhelníku. Nyní bychom měli být schopni s dostatečným množstvím známých údajů jednoduše dopočítat onu vzdálenost objektu Z.

Které z údajů tedy ve chvíli výpočtu již známe? Prvním je vzájemná vzdálenost kamer v obrázku značená jako b. Další známou hodnotou pak je ohnisková vzdálenost kamery f a nakonec velikost disparity tu určí algoritmus stereo korespondence. Z těchto tří údajů a vztahů popsaných v předešlém odstavci nyní můžeme odvodit vzorec pro výpočet vzdálenosti pozorovaného objektu.

Algoritmus určí disparitní vzdálenost $x_l - x_r$ tato vzdálenost je podobná vzdálenosti b. Poměr těchto vzdáleností považujeme za konstantu podobnosti uvažovaných trojúhel-

níků. Známe-li tuto veličinu jednoduchým vynásobením s ohniskovou vzdáleností f. Dostaneme vzdálenost objektu P označenou Z.

Definice 3.1 ***Formální definice disparity VSTUP:*** *Jasová složka referenčního obrazu $p_l(x, y)$ a jasová složka cílového obrazu $p_r(x, y)$.* ***VÝSTUP:*** *Mapa disparitních vzdáleností, určující korespondenci mezi oběma obrazy $D(x, y)$ je definována rozdílem souřadnic x dvou korespondujících bodů $p_l(x, y)$ a $p_r(x, y)$.*

$$D(x, y) = x_{p_l} - x_{p_r} \tag{1}$$

Věta 3.1 *Vzdálenost Z objektu P od ohniska kamer je definovaná jako podíl vzdálenosti kamer b a disparitní vzdálenosti D vynásobený ohniskovou vzdáleností f.*

$$Z = \frac{b}{D} f \tag{2}$$

Tento výpočet je ovšem zatím bezrozměrný jistě tušíme, že vzdálenosti se budou udávat v metrech, ovšem v reálném světě není nic ideální. Tím narážím hlavně na disparitní vzdálenost. Ta trpí největší nepřesností. Velikost vzájemné vzdálenosti kamer b a ohniskové vzdálenosti f můžeme zjistit velmi přesně. Disparitní vzdálenost ovšem měříme na námi zvoleném modelu. Ten není nikdy dokonalý a je limitován hlavně technickými parametry kamer vytvářejících množiny záznamů Π_l a Π_r. Směrodatným parametrem kamer je v tomto případě hlavně jejich rozlišení. Čím větší rozlišení máme k dispozici, tím přesnější výsledky můžeme získat. Kamery totiž mohou zaznamenat pouze omezený počet bodů takzvaných pixelů. Dnes se hodnota v běžných kamerách pohybuje okolo 10Mpx pro srovnání rozlišení lidského oka se udává až 137Mpx.

Pixel v digitální podobě ovšem nemá velikost v reálném světě pouze hodnotu. Plocha jednoho pixelu je vždy vztažena ke konkrétní kameře. Dnešní mikročipy se vyrábí technologií $32\ \mathrm{nm}$. Opět pro srovnání velikost jednoho receptoru v lidském oku se pohybuje od $1.5\ \mathrm{nm}$ do $3\ \mathrm{nm}$. Dnešní technika tudíž není limitována rozlišením optického senzoru ale hodnotou, která se za rozlišení často zaměňuje. Jedná se o dpi neboli počet bodů na palec. Například v teleskopech Pan-STARRS [11] je senzor s rozlišením 1,4Gpx avšak rozměry tohoto senzoru jsou také enormní a jedná se totiž o čtverec o hraně přibližně 40cm. Lepší představu o velikosti tohoto senzoru získáme z obrázku 18. Z důvodu zjednodušení se zavádí různé modely popisující obrazovou informaci.

Jedním ze známých je takzvaný pin hole model. Ten si lze představit jako dírkovaný list papíru nebo jemné síto, přes které se díváme. Využití tohoto modelu je velmi výhodné obzvláště vezmeme-li v potaz formát, v němž dnešní kamery zaznamenávají obraz. Jedná se totiž o rastrovou reprezentaci grafiky skládající se právě z jednotlivých pixelů. Každý pixel popisuje specifickou část obrazu. Popisovaná oblast odpovídá velikostně jednomu prvku snímacího senzoru a má své unikátní souřadnice x a y. Což přesně odpovídá modelu pin hole na obrázku 15. Tento model ovšem není dokonalý, proto je na obrázku 16 vyobrazen vztah mezi velikostí bodu v modelu a jemnosti rozlišení vzdálenosti objektu. Jak je vidět v předchozím modelu z obrázku 15 jsme si zjednodušili život a místo trojúhelníku $\triangle ADE$ zobrazeného na obrázku 1 jsme uvažovali pouze přímku a to těžnici

t_a trojúhelníku $\triangle ADE$. Chyba R se bude zmenšovat s rostoucím rozlišením dpi na druhou stranu ovšem poroste časová i paměťová náročnost výpočtu. Je třeba zvolit rozumný kompromis mezi přesností a výkonem. Tento jev je také důvodem proč se zdají velmi vzdálené objekty v záznamech statické tudíž disparitní vzdálenost je nulová, což dobře ilustruje obrázek 16b.

Problémů je, však mnohem více zmiňme problém překrytí vyobrazený na obrázku 17. Spočívající hlavně v neprůhlednosti většiny pozorovaných objektů. To způsobuje ztrátu informace o objektech umístěných za objektem pozorovaným. Na obrázku 17 jsou černou barvou vyznačeny slepá místa, o kterých nemáme žádné vizuální informace. Šedě jsou pak značeny oblasti, o kterých máme částečné informace, tudíž jsou viditelná pouze z jedné kamery. Dalším častým problémem je naopak průhlednost objektů. Ty mohou zůstat také zcela nepovšimnuty. To je velký problém vzhledem k možným důsledkům špatné analýzy prostoru. V těchto dvou případech řešení spočívá v zapojení inteligence a jiných obrazových vodítek. Tyto vodítka se využívají, mimo jiné k oklamání lidského zraku příklady naleznete v příloze A. Speciálně klamy na obrázcích 12b a 12e stojí za pozornost. Pracují totiž s doplňováním informace v případě 12e víme krychli za pomoci černých rohových bodů. Klam 12b oproti tomu lze implementovat dvěma různými způsoby. Simulovaní těchto procesů patří ovšem spíše do oblasti umělé inteligence. Dalším častým problémem je deformace optické soustavy to může mít za následek nutnost úpravy vztahů nebo znehodnoceni obrazové informace a v extrémním případě i znemožnit fungování stereoskopických algoritmů.

Nyní již víme jaké údaje a vztahy je nutno znát abychom vůbec mohli využít výsledky stereo korespondenčních algoritmů. Proto si pojďme nyní přiblížit samotný problém počítačové stereo korespondence a následně podrobněji popsat způsoby jeho řešení.

4 Stereo korespondence

Samotný problém stereo korespondence tedy hledaní souvisejících obrazových bodů ve dvou a vice scénách je až NP-těžký a to podle zvoleného typu řešení. Podrobnějším popisem metod řešení problému stereo korespondence za pomoci počítače se zabývá kapitola 5. Jednotlivé zkoumané záznamy scény jsou většinou všechny zachycený ve stejný čas, a tudíž v ideálním případě zachycují stejné objekty z různých uhlů.

Pak záleží na geometrickém uspořádání kamer. Toto uspořádání musí být přesně dané nebo zde musí být možnost jak přesně zjistit jeho parametry v okamžiku zachycení scény. Podrobněji se tomuto tématu věnuje předcházející kapitola 3. Počítačové řešení stereo korespondence má však i své úskalí. Ty se však výrazně liší od vad a slabin lidského zraku. Hlavně se jedná o tyto čtyři problémy .

1. **Šum**. Jedná se o různé rušivé elementy. Podmínky pořizovaní scény nejdou vždy dokonalé. Rozdíly v osvětlení scény z různých úhlů pohledu. Neostrost nebo špatné zaostření kamery tento problém se vyskytuje i u lidí v podobě krátkozrakosti nebo dalekozrakosti a řeší se brýlemi popřípadě čočkami. Nebo vadou či nedokonalostí snímacího senzoru obraz nemusí byt kompletní proto by měl být ideální algoritmus stereo korespondence dostatečně robustní, aby se s těmito problémy dokázal vypořádat.

2. **Oblasti bez textury**. Jedná se o oblasti, které jsou vyplněny jednotnou hodnotou tedy barvou. Tento jev dokáže zmást i člověka. Například si představme dokonale osvětlenou bílou plochu. Když neuvidíme její okraje, nebudeme ani schopni přesně určit sklon nebo vzdálenost. Ovšem člověk si zde pomůže ať už změnou pozorovací pozice nebo zkušeností. Stroj tyto možnosti mnohdy nemá. I když také zde je vidět také znatelný pokrok vpřed.

3. **Hloubkové nesrovnalosti**. Tímto pojmem jsou míněný náhle změny vzdálenosti objektu od kamery. Ty mohou vzniknout různými způsoby. Například se jedná o rohy, osamocené objekty hlavně menšího rázu. Pro představu se jedná například o tenisový míček, stojan kamery nebo různé sloupy a kabely. Zde je lidský zrak napřed vzhledem k jeho schopnosti vypomoct si zkušeností a jemným rozlišením jednotlivých předmětů. V tomto rozlišení hraje vekou roli práce s barvami.

4. **Problém překrytí**. Tento problém spočívá ve skutečnosti, že některé části scény jsou z jednoho úhlu vidět a z jiného ne. My můžeme s jistotou určit polohu pouze takových objektu, které vidíme aspoň dvěma pohledy z různých uhlů. Podrobněji jsme si tento problém vysvětlili v předcházející kapitole.

Mimo těchto hlavních problémů, které mají poměrně snadné řešení, jsou zde i jiné obtížněji řešitelné problémy. Jedná se hlavně o průhledné materiály, zrcadla nebo stíny. Mezi zajímavé problémy patří také vlnění horkého vzduchu, které můžeme spatřit třeba nad zapálenou svící. Tento jev v extrému dokáže způsobit fata-morgánu a za určitých podmínek může být poměrně nebezpečný. Tyto exotické optické problémy se řeší pouze

velmi obtížně. Většinou za pomoci jiných technik než je pouze jednoduché optické vnímání, které v těchto případech selhává. Podobné problémy se dají řešit různými způsoby. Někdy se uplatňují algoritmy využívající zkušenost a pravděpodobnost těmi se však již blíže zabývat nebudeme, jelikož bychom se dostali na pole umělé inteligence a ta není tématem této práce.

Další možností je využití jiných způsobů určení vzdálenosti. Tím jsou především myšlena vzdálenostní čidla dnes běžně užívaná v automobilech. Samozřejmě existuje nepřeberné množství získávání dodatečných informací k řešení našeho problému. Za všechny jmenujme laserové měřiče vzdáleností, které patří k nejpřesnějším metodám.

Úkolem této práce je ovšem rekonstrukce scény s využitím pouze obrazové informace. K tomuto účelu využijeme hlavně algoritmus anglicky zvaný Belief Propagation česky se tento název dá přeložit, jako propagace důvěry což jej přesně vystihuje. Jedná se velmi efektivní algoritmus detailně popsaný v kapitole 7. Ovšem k dosažení uspokojivých výsledků je nutné využít různé techniky optimalizace. Hlavně v oblasti času zpracování. Tyto techniky si představíme v kapitolách 8 a 9 zabývající se popisem samotného algoritmu, jeho vylepšeními a následnou implementací.

5 Algoritmy stereoskopické korespondence

V kapitole 3 zabývající se epipolární geometrií jsme si popsali geometrické vztahy související se záznamem scény a její rekonstrukcí. Následně jsme si detailněji popsali problém stereo korespondence a způsob jakým můžeme zkontrolovat výsledky.

Nyní už víme jak využít disparitní mapy ke zrekonstruování scény. Ze vztahu definovaných v kapitole 3 vyplynula souvislost mezi vzdáleností objektu od pozorovatele a jeho posunem v jednotlivých pohledech záznamu neboli disparitou. Jak již bylo zdůvodněno tento posun je pouze vodorovný a zpravidla se při hledání postupuje z levé do pravé strany. Což platí v případě, kdy uvažujeme levý obrázek jako referenční. Algoritmy provádějící toho hledaní vyhledávají takzvanou stereo korespondenci odtud název této práce.

Schopnost napodobit lidský zrak zde ovšem dostává trhliny. Vyvstává totiž otázka jak imitovat procesy probíhající v mozku. Ten je schopen reagovat na podněty běžně v čase 0,3s . Doba zpracování obrazové informace je ovšem ještě kratší. Důvodem je způsob měření tohoto času. To probíhá od zobrazení a po reakci člověka tím je ovšem zpravidla myšleno zmáčknutí tlačítka. Tudíž se do celkového času započítává také doba nutná k reakci svalů. I přes tento hendikep k dosažení podobného času zpracování v počítačovém vidění bylo nutno jak použít dostatečně výkonný hardware, tak i vyvinout optimální algoritmus stereo korespondence.

Ovšem dosáhnout správného výsledku je možno mnoha způsoby a také velmi záleží na formátu vstupních dat. Algoritmy stereo korespondence se dají dělit podle dvou hlavních faktorů a to podle způsobu zpracování dat a podle formátu dat.

Obraz je v počítačích reprezentován několika způsoby dvěma nejčastěji používanými typy, pokud jde o záznam nebo uchování obrazu jsou takzvaná rastrová a vektorová grafika. Vektorové grafiky se ovšem využívá spíše k samotnému modelování scény ve 3D. Také často využívá v technických aplikacích například AutoCAD nebo dnes velmi populární SolidWorks. Problémem je ovšem způsobu záznamu. Ten odpovídá spíše pin hole modelu popisovaného opět v kapitole 3. Kdy se celkový obraz skládá z mnoha malých bodů neboli pixelů.

Algoritmy stereo korespondence se v dnešní době dělí hlavně na lokální 5.1 a globální 5.2 podle způsobu jakým pracují s daty. Dalším možným kriteriem je kvalita výsledků zde se uplatňuje dělení na husté a řídké. Podle toho jestli je výsledkem pouze disparita význačných bodů a hran v případě řídkých algoritmů. Ty se využívaly hlavně z důvodů nízké náročnosti na výpočetní výkon. Druhou možností jsou algoritmy husté. Ty ohodnotí všechny nebo většinu bodů v obrazu. V dnešní době se zpravidla využívají algoritmy husté vzhledem k tomu, že již nechceme mít pouze hrubou představu o podobě scény, ale zajímají nás také detaily. V mnoha aplikacích je velmi důležité přesné vnímání prostoru, jako příklad uveďme medicínu. V ní může mít omyl fatální následky.

Dalším důležitým znakem algoritmů stereoskopické korespondence je využití barvy. Aktuálně nejlepší algoritmy všechny nějak využívají barvu. Princip využití barevných informací si blíže popíšeme v kapitole 7. Algoritmy pracující pouze s černobílou informací nebo jasovou složkou obrazu zanedbávají značné množství obrazové informace, ve které barva hraje velkou roli. Zpravidla je s její pomoci možné velmi dobré rozlišení jed-

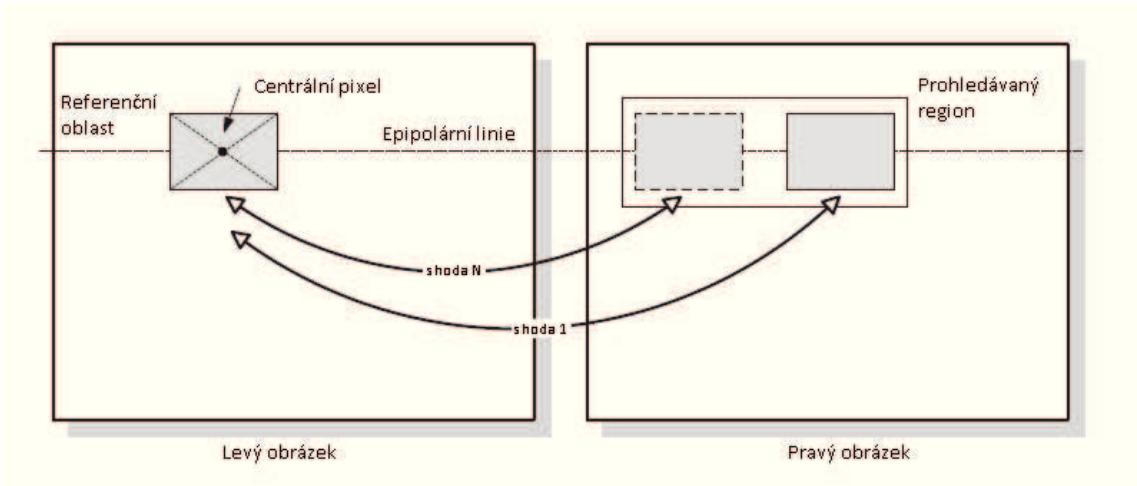
notlivých objektů v záznamu scény. Tato skutečnost nám pak umožní lépe určit disparitu v okrajích objektu.

5.1 Lokální algoritmy

Lokální algoritmy stereo korespondence také se někdy označují jako okénkové. Vyznačují se způsobem zpracování obrázku, ve kterém se porovnávají pouze malé oblasti neboli okna a to zcela nezávisle na sobě. Tyto algoritmy se dále dělí podle funkce ohodnocení shody jednotlivých oken. Vzorce jednotlivých funkcí obsahuje tabulka 2 v příloze C. Další možností dělení je podle způsobu zacházení s oknem jestli má statickou nebo dynamickou velikost a tvar. Variant a různých vylepšení těchto algoritmů bylo navrženo hodně.

Nejčastější vylepšení těchto typů algoritmů je právě zavedení dynamické velikosti okénka. Toto opatření má hlavně pomocí správně určit velikost disparity v oblastech bez textury. Kde by mohlo dojít k výraznému zkreslení v případě moc malého okna. Naopak v členitých oblastech příliš velké okénko způsobí zkreslení výsledku a důsledkem toho zaniknou detaily scény.

Tento typ algoritmů ovšem v dnešní době poměrně zaostává jak v přesnosti, ale hlavně v časech zpracování za globálními algoritmy. I přesto vzhledem velmi snadné paralelizaci můžou i tyto algoritmy poskytovat velmi slušné výsledky. Je však nutno využít správný hardware v dnešní době například GPU. V budoucnosti mohou být dostupné CPU s desítkami až stovkami jader. Tyto procesory dosáhnou mnohem lepších výsledku za pomoci lokálních algoritmů. Otázkou však je kolik jader v této době budou využívat GPU vzhledem k dnešnímu poměru jedna ku pěti mnohdy i více.



Obrázek 2: Princip lokální strategie zjišťování stereo korespondence [1]

Lokální algoritmy jsou také mimo jiné užitečné jako učebnicový příklad a to jak z důvodu snadné základní implementace tak i jasných výsledků. Postup algoritmu je jednoduchý a je zobrazen na obrázku 2. Zpravidla nezávisle na sobě prochází rastrový záznam obrazu po řádcích. Kolem každého pixelu pro levý pohled pak vytvoří okno

a porovná jeho obsah za pomoci jedné z funkcí uvedených v tabulce 2 s oknem okolo korespondujícího pixelu v pravém obrázku. Korespondující pixel leží z důvodu vztahů uvedených v kapitole 3 vždy na vodorovné přímce směrem doprava. V případě záměny levého a pravého záznamu je nutno postupovat opačně zprava doleva. Kdybychom nezměnili směr procházení, dostaneme jako výsledek pouze chaotickou změť hodnot. Okno se iterativně posunuje po této přímce. Přičemž jeho posun je ohraničen minimální disparitou, ta udává pomyslnou maximální vzdálenost objektu a zpravidla bývá nulová tudíž nekonečná vzdálenost. Další hranicí je pak maximální disparita udávající minimální vzdálenost pozorovaného objektu. Význam těchto proměnných ilustruje obrázek 16b. Hodnoty platné pro tento obrázek jsou 1 v případě minimální disparity a maximální disparita má hodnotu 7.

Po tomto ohodnocení se určí pravděpodobnost shody daných pixelů přičemž platí čím je rozdíl menší tím je větší pravděpodobnost shody. Z tohoto důvodů se zde často využívá principu „vítěz bere vše“ Výpočet disparity podle definice 3.1 je pak již jen formalita. Poté již následuje post procesing. Vyhlazení a oprava výsledků možností je celá řada. Celý postup lze shrnout do těchto čtyř kroků.

1. Ohodnocení shody pixelů.
2. Sloučení ohodnocení.
3. Výpočet disparity.
4. Zjemnění disparitní funkce

Tento typ algoritmu stereo korespondence byl popsán již poměrně dávno již na přelomu 80. a 90. Let minulého století. Poměrně podrobně se tomuto tématu věnuje kniha [1] nebo publikace pánů Kanadeho a Okutomiho [10].

5.2 Globální algoritmy

Globální algoritmy v dnešní době dosahují mnohem lepších výsledků nežli lokální algoritmy popsané v předešlé kapitole 5.1. Hlavním znakem globálních algoritmů je že na problém nahlížení jako na celek a z pravidla se řadí do skupiny hustých algoritmů. Postup výpočtu je odlišný než u lokálních algoritmů.

Globální algoritmy totiž pracují na principu hledání minimální ceny funkce ohodnocení disparity celého obrazu. Matematici v této souvislosti používají termín globální optimalizace. Funkce ohodnocení shody se v případě globálních algoritmů skládá z více proměnných.

$$E(f) = \alpha E_{DATA} + \beta E_{SMOOTH}$$

První proměnou je E_DATA reprezentující datovou složku. Tedy určitým způsobem vypočítaná data ze stereoskopického páru obrázků. K tomuto účelu lze využít funkce z přílohy C. Další složkou je takzvaná funkce linearity E_{SMOOTH} udávající pravděpodobnost shody hodnoty disparity s předchozími hodnotami. Každá z těchto proměnných je

pak následně přidělena cena neboli důležitost dané proměnné vůči ostatním. Minimalizační funkce se může skládat také z více než dvou proměnných. Zde se právě dostávají ke slovu další proměnné například již zmiňovaná barevná příslušnost k určité oblasti. Lze využít také jiných vlastností pixelů. Další důležitou součástí každé proměnné je její váha. Neboli koeficient udávající vý-znamnost proměnné. V našem vzorci jsou reprezentovány řeckými písmeny. Hodnota těchto kvocientu se určuje empiricky a následně zůstává zpravidla neměnná.

Ovšem ne všechny globální algoritmy dosahují dobrých výsledku. Mezi dávno překonané patří metoda simulace žíhání. A to hlavně z hlediska času zpracováni. Tato metoda je také poměrně nepřesná z důvodu využití náhody. Ovšem pořád umožňuje dosáhnout výsledku v lepším čase než lokální algoritmy na principu brute force. Princip je velmi prostý a vychází stejně jako název metody z metalurgie. Detailnější popis lze nalézt v diplomové práci Zdeňka Vašíčka [12].

Tato metoda se ovšem nepoužívá vzhledem k časové náročnosti dosažení kvalitního výsledku. Například práce [13] ukazuje jasnou převahu grafových řezů.

5.2.1 Graph Cuts

Grafové řezy jsou dalším velmi úspěšným přístupem k řešení problému stereo korespondence v obrazech a řadí se do skupiny globálních algoritmu. Sestavují totiž cenovou funkci o třech proměnných.

$$E(f) = E_{DATA} + E_{SMOOTH} + E_{OCC}$$

První proměnnou je E_{DATA} reprezentuje stejně jako v algoritmu BP datovou složku cenové funkce. K tomuto účelu lze využít funkce v příloze C nebo způsoby popsané v kapitole 9.1.1. Další je již zmíněná proměnná hladkosti E_{SMOOTH} reprezentující vztah vůči sousedním pixelům. Poslední proměnnou je cena překrytí E_{OCC}.

V první fázi algoritmu se jednotlivé body sdruží do skupin. Každá skupina zpravidla odpovídá objektu ve scéně a ten by také měl mít svou specifickou disparitu. Z této úvahy vyplývá, že k takzvanému labelingu neboli značkování můžeme ve finální fázi využít právě hodnotu disparity. Příslušnost pixelu ke skupině se ovšem nejdříve určuje na základě hodnoty rozdílu intenzit sousedních pixelů. Jednotlivé skupiny však mají z počátku pouze abstraktní hodnotu. Tato hodnota se následně s pomoci minimalizace cenové funkce převedena reálnou hodnotu vypočtené disparity.

Algoritmus k tomuto účelu využívat různé postupy s teorie grafů mezi důležité jmenujme barvení grafů nebo výpočet maximálního toku proto název grafové řezy. Tento algoritmus dosahuje mnohem přesnějších výsledků oproti dynamickému programování a to hlavně z důvodu využití souvislostí mezi řádky. Ovšem časová složitost grafových řezů je poměrně značná v určitých případech až v řádu minut, což značně limituje jejich využití. Z tohoto důvodů se tento přístup využívá zpravidla v kombinaci s jinými. Tento fakt dokazuje webová stránka [28]. Proto je tématem mé práce efektivnější algoritmus Belief Propagation.

Případné zájemce mohu odkázat na diplomovou práci Václava Lukáše [14], tato práce se grafovými řezy zbývá do detailů a její součástí je také implementace tohoto algoritmu.

5.2.2 Belief Propagation

Algoritmus Belief propagation nebo také česky propagace důvěry je založen na principu iterativního zasílání zpráv mezi jednotlivými body.

Princip tohoto algoritmu byl popsán roku 1982 v publikaci [17] a sloužil k výpočtu marginalu v Bayersových sítích. V našem případě BP ale využijeme pro výpočet pravděpodobné hodnoty v MRF.

Algoritmy založené na tomto principu dosahují poměrně dobrých výsledků. Podrobněji se budeme tímto druhem algoritmu zabývat v kapitole 7.

5.3 Ostatní metody

Některé algoritmy stereo korespondence nelze striktně zařadit do skupiny lokálních nebo globálních algoritmů. Tyto algoritmy zpravidla využívají podobnosti, návaznosti nebo jiných vztahů mezi jednotlivými hodnotami. Do této skupiny patří všechny algoritmy, které neprocházejí obrázek lineárně jako je tomu u lokálních algoritmů a zároveň nesestavují globální funkci k následné minimalizaci. Nebo algoritmy nepracující s klasickou intenzitou pixelu. Takovouto metodou je například metoda semiglobálního párování. Ta pracuje na principu vzájemné informace mezi sousedícími pixely. Myšlenka je z pravidla podobná jako v případě metod popsaných v předchozích kapitolách. Hlavní změnou v tomto případě je práce se vztahy mezi pixely nikoli s jejich intenzitou.

Tento přístup má řadu výhod. Vůbec jej neovlivní například invertování části obrázku nebo jeho zesvětlení popřípadě ztmavení. Zatímco složitost zůstává na úrovni algoritmů založených na porovnání intenzity pixelu. Tato metoda může dosahovat velmi dobrých výsledků, jak ukazuje práce [15].

Blíže si však popíšeme mnohem používanější metodu dynamického programování a to ze dvou důvodů. Tato metoda je totiž podobně jako lokální algoritmy popsané v kapitole 5.1 slouží často jako učebnicový přiklad řešení problému stereo korespondence. Druhým důvod pak je skutečnost že algoritmy semiglobálního párování jsou speciálním případem algoritmu dynamického programování.

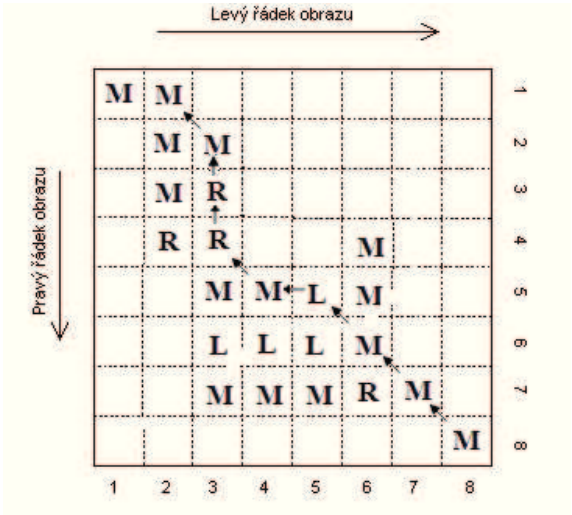
5.3.1 Dynamické programování

Tato metoda byla poprvé popsána Richardem Bellmanem roku 1940 název této metody, ovšem vůbec neodpovídá jeho principu. Tím je Bellmanův princip optimality „Podstrategie optimální strategie je opět optimální“.

Dynamické programování stále využívá okénko k ohodnocení shody mezi dvěma pixely. Ovšem postup průchodu algoritmu se velmi liší. Zatímco v případě lokálních algoritmů se neberou v potaz předešlé výsledky. Dynamické programování tyto výsledky bere v potaz a často jsou dokonce nutné k dosažení celkového výsledku. Proto se všechny

mezivýsledky ukládají do paměti. Tím sice dochází k nárůstu paměťové složitosti algoritmu, ovšem tento nárůst není kritický. A uchovávané informace nám pomůžou velmi zredukovat čas zpracování algoritmu.

Dynamické programování je bezesporu složitější než lokální algoritmy nebo simulace žíhání ovšem i přesto dosahuje lepších časů a zpravidla i výsledků. Při splnění několika podmínek a za pomoci mírného zjednodušením modelu. Lze problém hledání korespondence pro celý řádek převést na problém hledání nejkratší cesty v grafu. Touto vlastností by se tento algoritmus dal nazvat pologlobální z důvodu minimalizace celkové cenové funkce stejně jako to provádějí globální algoritmy ovšem pouze pro jeden řádek. Tohoto faktu však lze s úspěchem využít ke zkrácení doby výpočtu zavedením principů paralelizmu. Další možnou optimalizací je omezení výpočtu pouze na možné cesty. Cesta se hledá za pomocí průchodu vpřed. Následně se pak nalezená cesta ověří průchodem vzad. Výsledek tohoto postupu odpovídá disparitní mapě pro jeden řádek a je zobrazen formou matice na obrázku 3. M jsou značeny pixely shodné. R odpovídá pixelům nacházejícím se pouze v pravém obrázku a L pixelům nacházejícím se pouze v levém obrázku.



Obrázek 3: Disparitní mapa pro jeden řádek vypočtená za pomocí algoritmu dynamického programování vyjádřená maticí.

Výsledná disparitní mapa takovéhoto postupu ovšem trpí vadami. Vzhledem k předpokladu hladkosti disparity dělají algoritmům založeným na dynamickém programování problémy hlavně úzké předměty v popředí scény. Takovéto předměty se za určitých okolností mohou zcela ztratit nebo způsobí poměrně velké rušení v horizontálním směru.

Proto Post procesing hraje u dynamického programování velkou roli. Často se také používají dva průchody algoritmem. Jednou se vezme jako referenční levý obraz a podruhé pravý výsledná disparita se pak zprůměruje. Možností je mnoho stejně jako v případě lokálních algoritmů. Podrobným popisem této metody se zabývá práce [16].

6 Způsoby hodnocení a srovnání algoritmů

K objektivnímu hodnocení a srovnáni jednotlivých druhů algoritmů je nutné mít k dispozici, jak testovací data, tak i definované metriky srovnání, pomocí kterých proběhne porovnání vlastností jednotlivých algoritmů. Hlavním kriteriem je přesnost a správnost dosažených výsledků. Tuto vlastnost vyjadřuje procento špatně ohodnocených pixelů. Druhým neméně důležitým parametrem je doba zpracování dat neboli složitost algoritmu. Trošku v pozadí zájmu je paměťová náročnost jednotlivých algoritmů ovšem zcela neprávem.

Postupem vývoje začala být otázka srovnání různých typů stereo korespondenčních algoritmů kriticky důležitá. Proto vznikla stránka vision.middlebury.edu [28]. Ta si klade za cíl poskytnout soubory testovacích dat a umožnit srovnání algoritmů spadající do kategorie počítačového vidění mimo jiné také algoritmů stereo korespondence. K dispozici je několik desítek testovacích souborů dat. Také je možnost přihlásit svůj vlastní algoritmus a získat tak srovnání s více než stovkou různých implementací stereo korespondenčních algoritmů.

První důležitou komponentou nutnou k ohodnocení algoritmů stereo korespondence jsou testovací data.

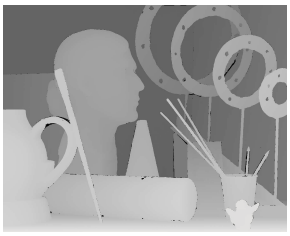
6.1 Testovací data

V počátcích zkoumaní a vývoje na poli počítačového vidění si každý tým většinou vytvářel vlastní soubor testovacích dat. K vytvoření testovacích dat se zpravidla využívají různé techniky založené na projekci předem známých obrazců na scénu. Následně se za pomoci úrovně deformace těchto promítaných obrazců určí disparitní mapa scény. Zájemcům o podrobný popis této metody lze doporučit publikaci High-Accuracy Stereo Depth Maps Using Structured Light [30].

Podobný přístup využívá také laserový paprsek. Ten ovšem nepromítá kompletní obrazec, ale osciluje na přímce. Ta se posunuje po scéně a její deformace opět určí úroveň disparity. Podobný princip využívá třeba počítačový skener nebo některé typy čteček čárových kódu. Tyto metody jsou poměrně přesné a v některých případech je vhodnější spíše jejich využití namísto rekonstrukce scény z obrazové informace za pomoci algoritmů stereo korespondence.

Například uveďme systém pohybového ovládáni Kinect od společnosti Microsoft. Tento systém právě s úspěchem využívá kombinaci obrazové informace a informaci získaných za pomoci projekce infračervené sítě.

Aktuálně se využívá formát testovacích dat zobrazený na obrázku 4. Každý soubor testovacích dat se skládá ze dvou pohledů levého a pravého. Dále je pak poskytnuta takzvaná ground truth což je vlastně přesná disparitní mapa získaná za pomoci již dříve zmíněných metod. Čtyři hlavní soubory dat [9, 22, 23, 24] naleznete v příloze D. V tabulce 1 jsou uvedeny parametry nastavení algoritmu, aby bylo srovnání objektivní. Uvedené parametry jsou globální a dokáží velmi ovlivnit dosahované výsledky všech druhů algoritmů. Jediným parametrem, který by nás mohl zmást je násobitel. Ten reprezentuje konstantu, kterou se vynásobí výsledek vypočtený algoritmem. Využívá se z důvodu lepšího


(a) Ground truth


(b) Levý pohled


(c) Pravý pohled

Obrázek 4: Příklad datového setu

Název datového setu	Minimální a maximální disparita	Násobitel
Tsukuba	0..15	16
Venus	0..19	8
Teddy	0..59	4
Cones	0..59	4

Tabulka 1: Parametry stereo korespondenčních algoritmů pro jednotlivé datové sety

využití rozsahu černobílého obrázku s hloubkou jeden byte. Což odpovídá celočíselným hodnotám z intervalu $< 0, 254 >$. Tento typ obrázku se používá ke grafické reprezentaci disparitní mapy. Některé druhy algoritmů pak mají i své specifické parametry. Ty bývají zpravidla společné pro určitou skupinu. Tímto tématem jsme se zabývali v předchozí kapitole 5.

6.2 Metriky srovnání

Samotné srovnání algoritmu pak lze provést několika způsoby. První a nejjednodušší je nahrát své výsledky na uvedený web [28]. Druhý způsob je vlastní srovnaní na základě Groud truth poskytnutého v testovacích datech. Ovšem co je vlastně výsledkem takovéhoto srovnání? Často se pracuje s procentem spatně nebo správně ohodnocených pixelů. V literatuře [9] se pak zavadí tyto pojmy.

Definice 6.1 *R, střední kvadratická chyba mezi vypočtenou disparitní mapou a skutečnou hloubkou.*

$$R = \sqrt{\frac{1}{N} \sum_{(x,y)} (d_c(x,y) - d_t(x,y))^2} \tag{3}$$

Definice 6.2 *B, poměr chybně detekovaných pixelů k celkovému počtu pixelů (se specifikovanou tolerancí).*

$$B = \frac{1}{N} \sum_{(x,y)} |d_c(x, y - d_t(x,y))| > \delta \tag{4}$$

Na zmíněném webu je pak konečným hodnocením takzvaná úroveň neboli anglicky rank což je poměr mezi správnosti dosazených výsledků a rychlosti zpracování. Výsledky

těchto srovnání ovšem nejsou vždy úplně objektivní. Stereo korespondenční algoritmy totiž dokáže zmást mnoho různých efektů a anomálií. A tak se poměrně často využívá takzvaného post procesingu. Jeho součástí jsou různé procesy vyhlazování a korekce výsledků. Ty pak zpravidla více odpovídají realitě, i když je zde možnost zanesení chyby. Proto je také tolik různých variací stereo korespondenčních algoritmů.

Další možností jak dosáhnout detailního srovnání algoritmů je rozdělit scénu a srovnávat pixely podle určitých podmínek například pouze body v oblastech s diskontinuitami, v oblastech překrytí nebo v oblastech bez textury. Ideální algoritmus by měl vždy dodat optimální výsledek.

7 Belief propagation

Algoritmus Belief propagation jsem se rozhodl implementovat a následně také optimalizovat. Proto je jeho popisu věnována tato kapitola. Tématem optimalizace se pak zabývám v samostatné kapitole 8.

Momentálně nejúspěšnější implementace algoritmu BP jsou AdaptingBP [18] a DoubleBP [19]. Tyto dva algoritmy se v době vzniku této práce nacházejí v žebříčku zveřejněném na webu [28] mezi prvními pěti. Důležitým faktorem těchto algoritmů je mimo jiné využití metody detekce ploch na základě barevné hodnoty pixelu. Následně se těmto plochám přiřadí hodnota disparity za pomoci propagace zpráv. Tento úkol lze samozřejmě řešit i jinak než pomoci algoritmu BP. Tyto alternativy jsou však hlavním tématem kapitoly 5.

BP je globálním algoritmem, tudíž musí vytvářet cenovou funkci a tu následně minimalizuje. V případě BP za pomoci zasílaní zpráv.

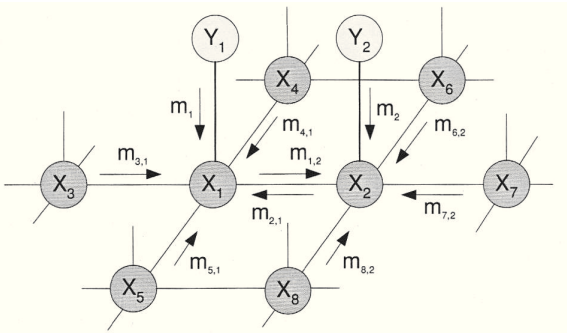
$$E(f) = E_{DATA} + E_{SMOOTH}$$

E_{DATA} pro data a E_{SMOOTH} jako vliv okolí jako všechny globální algoritmy. Vzorové algoritmy BP využívají ke zpřesnění funkce E_{SMOOTH} barevné segmentace obrazu.

Nyní si popíšeme základní principy algoritmu. Ten probíhá v několika fázích a využívá poměrně specifické datové struktury.

7.1 Markovovy náhodná pole

Základní strukturou využívanou v algoritmu BP jsou Markovovy náhodná pole dále již jen MRF. Příklad MRF zobrazuje obrázek 5. Tato struktura je založena na neorientovaném grafu s ohodnocenými vrcholy. Každý vrchol pak má definované takzvané skryté proměnné značené šedou barvou a reprezentuje jednu náhodnou proměnnou, v obrázku mají barvu bílou. MRF jsou dále doplněny o definici závislostí, ty jsou reprezentovány hranami grafu. Vrcholy, které nejsou navzájem spojeny hranou, jsou na sobě nezávislé.



Obrázek 5: Markovovo nahodné pole

Vezmeme-li v potaz pinhole model popsaný v kapitole 3 a fakt že pracujeme s rastrovou grafikou je aplikace MRF na tento problém poměrně snadná. Jednotlivé pixely ozna-

číme za náhodné proměnné a jakékoliv další informace budeme ukládat do náhodných a skrytých proměnných.

Další důležitou vlastností MRF je definice vztahu mezi závislými náhodnými poměnnými. Tyto vztahy jsou dále popisovány jako zprávy. Hodnota a formát zprávy se odvíjí od specifické implementace MRF.

7.2 Fáze algoritmu BP

Algoritmus BP můžeme rozdělit na několik hlavních fází, jejichž pořadí je neměnné.

1. Výpočet výchozích dat.
2. Iterativní posílaní zpráv.
3. Výpočet výsledku.

Prvním důležitým krokem je výpočet výchozích dat ty jsou reprezentovány pomocí MRF. Tato struktura nám umožňuje uložení všech potřebných údajů. Data mohou byt počítána mnoha způsoby. Zpravidla se využívají různé metody detekce rohů, hran a ploch. Přesto hlavním zdrojem dat zůstává rozdíl hodnot pixelu levého a pravého obrázku. Získání těchto informací zpravidla zabírá značnou část doby běhu algoritmu BP. A právě způsobem výpočtu výchozích dat se liší mnoho verzi BP.

Nejjednodušší formou výpočtu je jednoduchá absolutní hodnota rozdílu dvou pixelů. Složitější funkce pracují pět s okénkem a mohou brát v potaz i jednotlivé barevné kanály. Mnou zvolená funkce pro výpočet základních dat je uvedena v kapitole 9 zabývající se samotnou implementací algoritmu BP. Podstatnou změnou oproti lokálním algoritmům je forma aplikace minimální a maximální hodnoty disparity. V případě globálních metod se tyto dvě hodnoty zpravidla neuplatňují při samotném výpočtu ale právě při výpočtu zdrojových dat. Algoritmus BP není výjimkou. Pro každou hodnotu disparity vytvoří jednu kompletní vrstvu náhodných proměnných. Odtud plyne poměrně značná paměťová složitost tohoto algoritmu.

Máme li vypočítány základní data. Můžeme přistoupit k iterativnímu zasílaní zpráv. Zprávy zasíláme v ideálním případě, dokud není problém vyřešen tuto informaci lze snadno získat hlavně v případě implementací BP s přívlastkem rychle konvergující. Blíže se tomuto tématu věnuje kapitola 8.1.2. Algoritmus BP je totiž ve své základní verzi poměrně náročný na výpočetní výkon. Hlavní problém nicméně tvoří paměťová složitost. To je způsobeno udržováním informací o značném množství náhodných proměnných. Tyto proměnné si navíc mezi sebou zasílají značné množství zpráv, které již nemají šanci ovlivnit výsledek. Z tohoto důvodu se využívají procedurální vylepšení algoritmu BP. Těmi se však zabývám v následující kapitole 8.

BP funguje hlavně z důvodu robustnosti mechanismu zasílaní zpráv. Existují dva základní typy zprávy. Při implementaci je nutno si jeden typ zvolit. Důvod je prosty tyto dva typy se nedají kombinovat.

- Maximální produkt.

- Sum produkt.

Maximální produkt zasílá dále zprávu s maximální hodnotou přicházející do odesílajícího vrcholu. Tudíž by se dalo říci, že vybírá nejhorší zprávu a tu posílá dále. Mohlo by se zdát, že název neodpovídá skutečnosti, ovšem musíme si uvědomit, že nejmenší rozdíl je ten nejlepší. Suma produktu jak už název naznačuje, provádí sumu všech příchozích zpráv a tuto hodnotu pak distribuuje dále. Z těchto dvou typů si lze libovolně vybrat. V případě typu sum produkt mohou hodnoty zprávy poměrně snadno přesáhnou maximální hodnoty menších datových typů.

Zprávy se zasílají iterativně v cyklech a každá náhodná proměnná dokáže v základu generovat čtyři zprávy. Ty odpovídají skrytým proměnným hodnoty, která je vygenerovala. Tato vlastnost je zajištěna vytvořením jedno pixelového rámečku kolem obrázku čímž se eliminují anomálie na okrajích. Navíc se zavádí vlastnost rozděleni pixelu do skupin podle iteraci většinou do dvou skupin na sudé a liché. Toto opatření nejenom zkrátí dobu průchod jednou iterací na polovinu, ale také zlepší efektivitu zpráv. V praxi se však zpravidla zpracovává najednou celý vektor náhodných poměrných reprezentující kompletní 3D informaci jednoho referenčního bodu. Tudíž za jednu zprávu lze považovat vypočet vektoru reprezentujícího vztahy s jedním ze sousedů aktuálně zpracovávaného bodu. Podrobněji se problémem výpočtu zpráv zabývám v kapitole 9.1.2.

Poslední fází algoritmu BP je výpočet výsledné disparity. Zde opět záleží na dané implementací BP. Tento výpočet není nijak složitý. Jednoduše se projdou hodnoty vypočtených dat. Data si můžeme představit jako kompletní záznam 3D scény. Pro každý pixel v tomto záznamu existuje odpovídající počet proměnných. Ty jsou seřazeny podle disparity, kterou reprezentují. Neboli by se dalo říci že výsledkem je hloubkový vektor daného bodu. Nám zbývá vybrat pouze tu nejpravděpodobnější hodnotu v tomto vektoru. Tou bude ta nejnižší a v ideálním případě bude vždy pouze jedna. Z tohoto důvodu se často využívá metoda „vítěz bere vše“ a k výpočtu stačí pouze jeden kompletní průchod daty. Hodnotou disparity pak bude délka vektoru o začátku po vítěznou hodnotu. Po tomto výpočtu může následovat ještě post procesing algoritmy založené na BP ovšem dosahují tak kvalitních výsledků že jejich další úprava je již zbytečná.

8 Optimalizace Belief propagation

Metod optimalizace algoritmu je mnoho vzhledem k jeho již poměrně dlouhému vývoji. Ať už se jedná o optimalizaci samotného algoritmu nebo o jeho efektivnější zpracování v podobě paralelizace. A to ať už na úrovni operačního systému, procesoru nebo grafického akcelerátoru. Jakožto u mnoha problémů předtím se jeví jako vhodné požít kombinaci několika způsobů optimalizace a to hlavně v návaznosti a závislosti jednoho způsobu na druhem což si přiblížíme v následujících kapitolách.

8.1 Procedurální vylepšení algoritmu BP

Jak již bylo zmíněno v předchozí kapitole 7 je časová složitost čisté implementace algoritmu BP poměrně značná. Z tohoto důvodu se do algoritmu BP zavádějí různá procesní vylepšení. Některá se už dokonce považují za standart, proto byla zmíněna již v předcházející kapitole. Díky těmto úpravám se velmi sníží časová složitost nebo paměťová náročnost algoritmu BP.

8.1.1 Datové uzávěry

Prvním druhem procesního vylepšení jsou datové uzávěry. Ty se zavádějí hlavně z důvodu úspory paměti. V praxi má totiž každý datový typ svou maximální hodnotu. Toto omezení nás na běžném formátu vstupního obrázku RGB-8 bitů nebo v případě černobílého obrázku nemusí až tak trápit. Ovšem záleží také na používaném typu zpráv. Typ Sum produkt bude mít tendenci dosahovat maximálních hodnot mnohem rychleji než Maximální produkt. Dalším důležitým faktorem je délka zpráv.

V případě dosažení nebo překročení maximální hodnoty datového typu program havaruje. Z tohoto důvodu je nutno této události předcházet. Opatření je možno využít několik prvním a nejjednodušším je zvolit dostatečně velký datový typ to ovšem nemusí být vždy možné.

Druhým je využití takzvaných datových uzávěrů. V praxi nás totiž nezajímají největší rozdíly ale ty nejmenší. Tudíž můžeme zavést konstantu nebo proměnnou, která bude udávat maximální rozdíl. Po překročení této hodnoty se bude rozdíl považovat za absolutní a nebude se již dále zvětšovat. Ve většině případů se velikost této hodnoty určuje empiricky.

Funkce datových uzávěru ovšem nespočívá pouze ve zpomaleni nárůstu hodnot a zabránění přetečení datových typů. Fungují také jako filtr k vyloučení extremních hodnot.

8.1.2 Uzávěry důvěry

Algoritmus BP ve své základní verzi posílá zprávy iterativně všem vrcholům. Počet iterací znatelně ovlivňuje kvalitu výsledku. Platí úměra čím vyšší počet iterací tím kvalitnější výsledek. Ovšem každá iterace neboli průchod všemi vrcholy v MRF je poměrně

časově náročná. Navíc o většině zkoumaných bodů můžeme s jistotou určit disparitu již při několika prvních průchodech.

Řešením problému takto vzniklých zbytečných zpráv je zavedení další konstanty reprezentující mez důvěry ve správnost daného výsledku. Pokud výsledek funkce důvěry překročí danou hodnotu. Bude problém disparity pro daný bod považován za vyřešený a další zprávy pro tento bod se již zasílat nebudou. Vzorec výpočtu hodnoty důvěry je věcí individuální implementace a hodnota konstanty se určuje opět empiricky.

Toto opatření se projeví hlavně obrovským snížením času výpočtu. A jedná se o jedno z nejpopulárnějších a reprezentuje jeden způsob dělení algoritmu BP. Jestliže implementace algoritmu BP využívá tohoto princip. Dostává přívlastek rychle konvergující. V anglické literatuře se užívá názvu „Fast converging“.

Implementace tohoto procedurálního vylepšení se mohou lišit hlavně ve formě výpočtu hodnoty důvěry. A přesným místem implementace v algoritmu. Výsledná funkce by měla být v každém případě stejná. Podrobnosti o mém řešení tohoto vylepšení se nalézají v kapitole 9.1.2.

8.1.3 Datová pyramida

Dalším z velmi efektivních přístupů k optimalizaci algoritmu Belief Propagation je datová pyramida. Hlavní princip spočívá v kompresi informace obsažené v MRF.

Komprese probíhá za pomoci kompresního okna, jak ilustruje obrázek 6. Provádí se v určitém kompresním poměru. Samotný výpočet komprimované informace je opět závislý na implementaci a budeme se jím podrobněji zbývat v kapitole 9.

Každé opakování procesu komprese vytvoří jedenu úroveň dat Belief Propagation. Tyto data se následně zpracují algoritmem BP úplně stejně jako data nekomprimovaná. Vypočtené výsledky se následně propagují proti směru komprese. Při tomto procesu mohou byt data opět dekomprimovány či jinak upraveny. Tudíž je nutno nejprve zpracovat úroveň s nejvyšší úrovní komprese. Následně se pokračuje následujícími úrovněmi. Dokud se tímto postupem nevrátíme opět na nekomprimovanou neboli výchozí úroveň. Tvořenou surovými neupravenými daty.

Tento přístup ilustrovaný obrázkem 6 je výhodný v tom že nám pro dosažení kvalitního výsledku stačí méně iterací v algoritmu Belief Propagation. Každá iterace přitom prochází jednou kompletně celé pole dat dané úrovně. Při konečné délce zpráv 25 což odpovídá 5 úrovním po 5 iteracích BP. Bude časová úspora značná. Tuto úsporu si můžeme dokonce vyčíslit. Ve výpočtu budeme uvažovat obrázek o velikosti 288x384 pixelů. Velikost kompresního okna zvolíme na 2x2 pixelů neboli kompresní poměr 2.

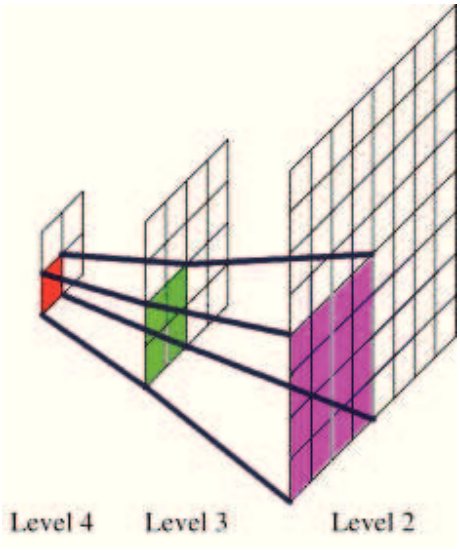
Příklad 8.1
Příklad vyčísluje úsporu kroků nutných k vykonání algoritmu BP v porovnání s hierarchickým algoritmem. Parametry pro výpočet jsou převzaty z předchozího odstavce.

$(384 * 288) * 25 = 2764800$ – 25 nekomprimovaných průchodu

$(384 * 288) * 5 + (192 * 144) * 5 + (96 * 72) * 5 + (48 * 36) * 5 + (24 * 18) * 5 = 736560$

736560 – kompletní průchod datovou pyramidou

$2764800 - 736560 = 2028240$



Obrázek 6: Datová pyramida

Oproti klasické implementaci jsme ušetřili 2 028 240 kroků. ■

Pokud se jedná o otázku paměťové náročnosti algoritmu. Ta se zvedá úměrně s počtem úrovní MRF neboli a počtem hledaných úrovní disparity. Ovšem v případě datové pyramidy není nárůst paměťové složitosti až tak velký. A vzhledem k počtu ušetřených operací se tuto dodatečnou paměť vyplatí obětovat. Čas zpracování navíc exponenciálně roste s velikostí obrázku tak i s počtem prohledávaných úrovní disparity. Proto je jakákoli úspora času zpracování velmi žádoucí.

Přístup datové pyramidy může mít i své varianty. Jako vhodná modifikace se jeví dynamická komprese. U menšího obrázku by bylo vhodné zvolit menší kompresní okno např. 2x2 aby se nám neztrácelo příliš mnoho z obrazové informace. Z druhé strany u velkých obrázků je možno zvolit kompresní okno větší než je 2x2 a tím docílit jak významného zrychlení algoritmu, tak i částečné úspory paměti. Hlavní myšlenkou tohoto vylepšení je určení kompresního poměru na začátku podle velikosti zdrojových obrázků. Během běhu algoritmu zůstává poměr konstantní. Toto řešení se jeví jako nejideálnější vzhledem k problémům u velkých obrázků. Dochází u nich velmi snadno k přeplnění paměti.

Dále se tímto principem optimalizace budeme zbývat v kapitole 9 při popisu mojí implementace algoritmu BP. Tento princip podobně jako v případě datových uzávěrů popsaných v předešlé kapitole 8.1.2 charakterizuje skupinu algoritmu BP, které ho využívají. Poznáme je podle přívlastku „hierarchické“ nebo anglicky „hierarchical“.

8.2 Paralelní zpracování

Většina algoritmů zpracovávajících grafická data umožňuje minimálně na některých místech využit principy paralelismu zpravidla, ovšem bývají možnosti mnohem rozsáhlejší.

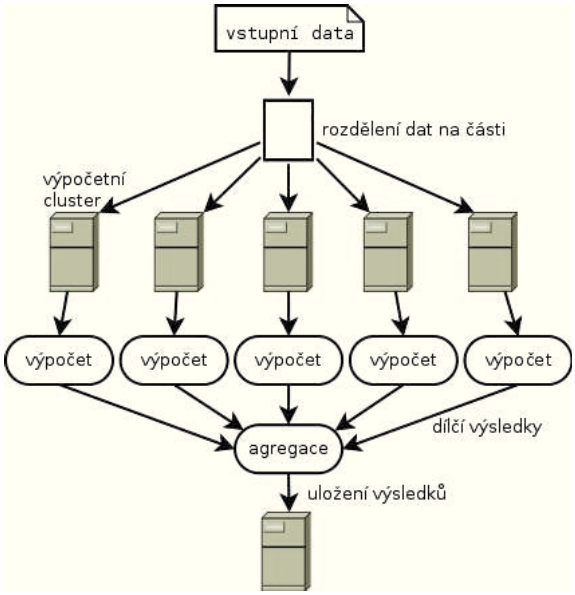
Hlavní podmínkou je, aby dílčí výpočty algoritmu byly na sobě nezávislé nebo alespoň minimálně. Tím je myšlena hlavně jasná hierarchie dat a postupu výpočtu. V dnešní době existuje mnoho způsobů jak docílit aby byly některé části algoritmu provedeny paralelně.

Myšlenka paralelismu spočívá hlavně v rozdělení velkého úkolu na vice menších a jejich následné hromadné zpracovaní. To provádí každá výpočetní jednotka nezávisle na jiné. Výpočetní jednotka, která provádí onen malý výpočet, se v každém případě liší. Může se jednat o jádro mikročipu nebo i celý počítač či grafickou kartu.

Vzhledem k tomu že hlavními požadavky na náš algoritmus Belief Propagarion je ne jen správnost výsledku, ale také čas jeho dosažení je vhodné použit některý nebo kombinaci níže zmíněných způsobů paralelizace. Pojďme si tedy nyní představit nejběžnější způsoby jak docílit paralelního zpracování.

8.2.1 Distribuované výpočty

Jednou z velmi účinných ale také velmi specifických technik paralelního zpracování jsou distribuované výpočty. V dnešní době se tento druh optimalizace využívá hlavně při řešení enormně velkých problému. V tomto případě je onou zmiňovanou výpočetní jednotkou právě celý počítač. Ten dostane po síti část dat děleného problému a po jejich zpracování odešle výsledky. Tento princip názorně zobrazuje obrázek 7.



Obrázek 7: Vizualizace principu distribuovaného výpočtu

Tento přístup má jistě své využití u velmi komplikovaných a rozsáhlých výpočtů, u kterých by dosazení výsledku bylo přinejmenším zdlouhavé a náročné pro jediný stroj. Druhou možností je získání velkého výpočetního výkonu za dobrovolného přispění velmi mnoha velmi malých výpočetních výkonů. Tím je myšlena možnost zapojit po internetu zahálející domácí počítače běžných uživatelů. Na internetu se dokonce formují týmy. Ty

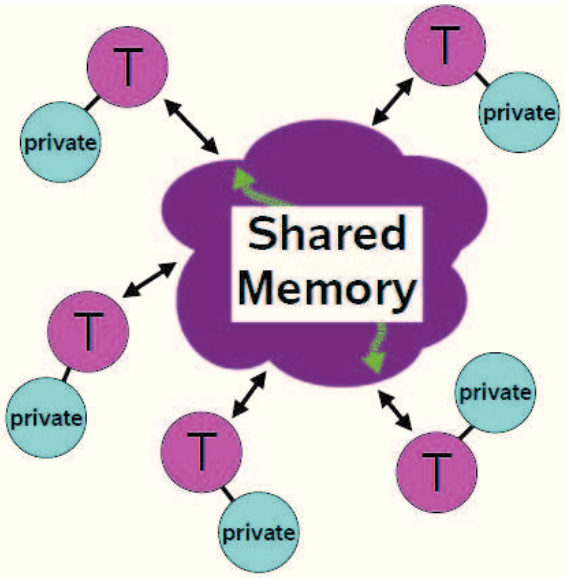
mezi sebou soupeři v rychlosti počítaní. Český tým má zkratku CNT [20] a může se k němu připojit každý. Stačí pouze přístup na internet a PC.

Princip distribuovaných výpočtu je však náchylný na chyby. Ty jsou hlavně v problému transportu dat, jelikož počítačové sítě nelze považovat za zcela spolehlivé a kdykoli může dojít k výpadku části dat z důvodu náhlé nedostupnosti počítačů, které tyto data zpracovávaly. A to ať už z důvodu poruchy sítě, výpadku proudu nebo poruchy samotného PC.

Dále mírně narůstá složitost a doba zpracování algoritmu o nutnou režii procesu distribuovaného výpočtu. Dobu zpracování ovšem nejvíce protahuje přenos dat po síti a proto je třeba pečlivě zvážit využití této techniky. V našem případě by byla vhodná pro zpracování dat ze stereoskopických teleskopů. Pro náš účel se však nehodí

8.2.2 Vlákna

Vlákna jsou jedním z nejjednodušších ale i nejrozšířenějších způsobů paralelizace. Můžou významně zkrátit dobu zpracovaní, ale jejich nadměrné využití může způsobit paradox nárůstu délky výpočetního času. To je způsobeno režií, která je nutná ke správě vláken a komunikaci mezi nimi. Vlákna jsou velmi vhodná pro naši implementaci Belief propagation hned z několika důvodů. Prvním je velmi snadné sdíleni dat mezi vlákny. Tato vlastnost je dána už ze samotného principu vláken. Ten byl navržen tak aby umožnil jednotlivým vláknům snadný přístup do globální paměti programu. Navíc má každé vlákno také svůj oddělený paměťový prostor. V něm si může ukládat lokální informace a mezi výpočty. Tuto strukturu ilustruje obrázek 8.



Obrázek 8: Zjednodušený model paměti pro vlákna.

Dnes již není nevýhodou ani závislost na operačním systému, kde se na Linuxu využívaly jiné knihovny než na Windows a tudíž bylo nutno optimalizovat algoritmus pro

každý operační systém zvlášť. Tato nevýhoda se projevovala hlavně nárůstem doby implementace a testování. Ovšem situace se v tomto ohledu výrazně zlepšila vzhledem k dobré implementaci standardu POSIX. Vlákna jsou vhodná především při zpracování velkých vstupních dat. Toto se dá ovšem prohlásit o každé formě paralelizace. U vláken můžeme přesněji říci, že jsou předurčena pro první úroveň optimalizace za pomoci paralelismu.

Vlákna ovšem přináší také jistá úskalí v podobě deadlocku, souběhu nebo nutnosti řešit kritické sekce. Bohužel tato práce neumožňuje detailnější popis všech těchto problému a způsobů jejich řešení. Hlavní příčinou tohoto faktu je obsáhlost této problematiky. Čtenářům lze na toto téma doporučit kapitolu z knihy Operating systems: design and implementation [21]. Ta detailně popisuje problematiku procesu a vláken a jejich paralelizace.

Dále se budeme vlákny zabývat, až v kapitole 9 ta se zabývá naší implementací námi zvoleného algoritmu Belief propagation. Vlákna při implementaci použijeme jako první úroveň optimalizace ale více až v již zmíněné kapitole 9.

8.2.3 Instrukce procesoru

Instrukcí procesoru existuje mnoho sad a verzí velcí výrobci přišli se svými specifikacemi. I přesto se dá říct, že výjimečné postavení mezi nimi má společnost Intel. Ta přivedla na trh vlastní sadu instrukcí MMX a SSE. Což jsou sady instrukcí samotného procesoru umožňující paralelizaci. Společnost Intel s tímto přístupem přišla 8. ledna 1997. Kdy představila procesor Pentium. Ten obsahoval zatím pouze instrukční sadu MMX vytvořenou z 57 instrukcí a z osmi 64 bitových registrů.

Ty to instrukce pracují na principu SIMD, tento princip jak již z anglického názvu vyplývá, využívá jednu instrukci na velké množství dat. Instrukční sada MMX však ještě neměla velký úspěch a to hlavně z důvodů malé nabídky procesorů, které by tyto instrukce podporovaly. Což se také projevovalo v nabídce programu vyvíjených s podporou těchto instrukcí. Tento stav se postupem času měnil. V tomto procesu Intelu výrazně pomohl konkurent společnost AMD uvedením procesoru K6 který podporoval MMX. Nedlouho poté, 7. května 1997, Intel představil Pentium II. Tím byly položeny pevné základy k podpoře vývojářů pracujících s instrukcemi MMX. Nyní je MMX standardem, který implementuje převážná většina výrobců procesorů určených pro osobní počítače.

Nicméně pro naše účely je instrukční sada MMX zcela nevhodná z prostého důvodu pracuje totiž pouze s celými čísly. Důvodem tohoto chování je sdílení registrů. Ty využívají stejný prostor jako pro registry určené pro práci s čísly v plovoucí desetinné čárce. Tento fakt znemožňoval míchání kódu zpracovávajícího desetinná čísla a instrukce MMX. Implementace našeho algoritmu bohužel využívá převážně datový typ float. Tudíž instrukce MMX nemůžeme využít ke zpracování navrženého algoritmu. Vzhledem k tomuto faktu bychom museli data pracně transformovat z desetinných čísel na celá popřípadě ještě po dokončení operace opět zpět na desetinná čísla. Tento proces by jednak zanášel do výpočtu chybu a také by prodlužoval dobu běhu algoritmu. Druhou snadnější cestou bylo navržení algoritmu pracujícího nad celými čísly. Naštěstí tyto problémy řešit nemusíme. Jednoduše použijeme jinou instrukční sadu.

Tou by mohla být sada instrukční 3DNow! společnosti AMD. Ta byla uvedena 28. května 1998 s procesorem K6-2. 3DNow! přidává k MMX dalších 21 instrukcí ty umožnily pracovat i s desetinnými čísly a to o délce 32-bitů což odpovídá datovému typu float do registru instrukční sady MMX se vlezou dvě hodnoty. Ve své době se jednalo o velký hit a procesory společnosti AMD dosahovaly oproti konkurenci skvělých výsledků. Ovšem tuto instrukční sadu a její následná rozšíření implementují pouze procesory společnosti AMD. Nicméně 3DNow! patři mezi jedny z nejpoužívanějších alternativ k instrukční sadě SSE a hlavně byla uvedena skoro o rok dříve než SSE.

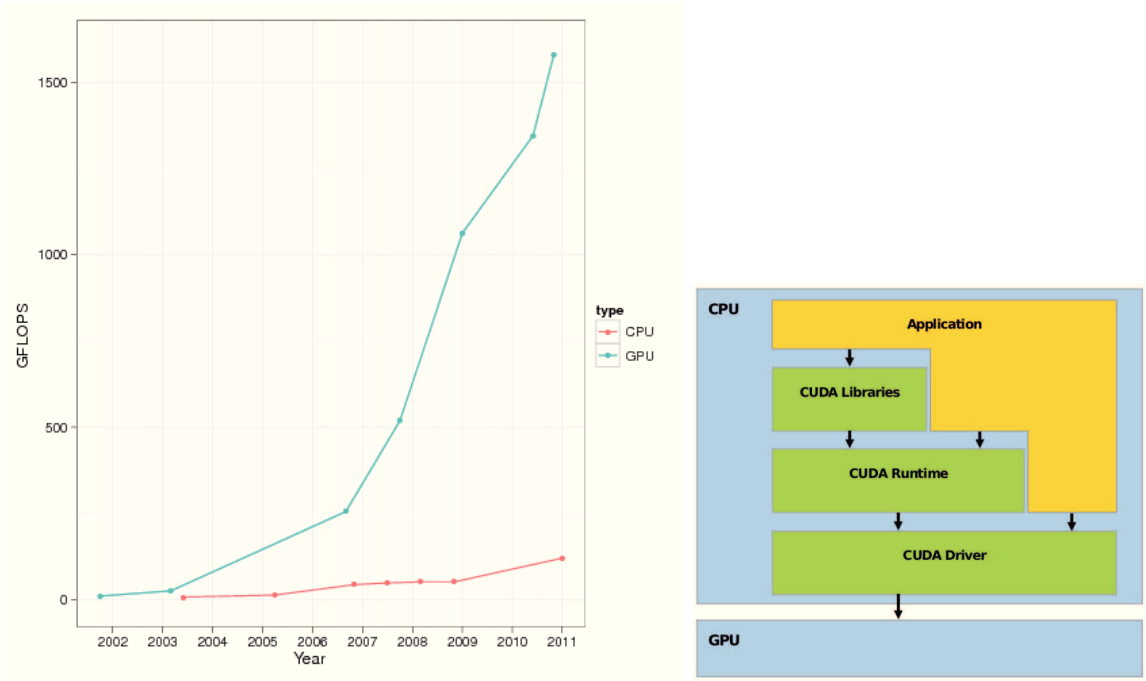
Instrukční sadu SSE uvedla společnost Intel až 26. února 1999 jako přímého nástupce instrukční sady MMX. Prvním procesorem s touto instrukční sadou byl Pentium III. Instrukční sada SSE oproti MMX obsahovala 70 nových instrukcí. Sada však opět musela počkat na masové rozšíření procesorů s její podporou první levné procesory Celeron s podporou SSE se objevily až v březnu 2000. Následně se roku 2001 přidala společnost AMD se svými procesory Athlon XP a Duron ty obsahovaly instrukční sadu 3DNow! Profesional což je vlastně 3DNow! a SSE dohromady. Instrukční sada SSE se ujala. Hlavně vzhledem k nově přinášeným možnostem. Umožnila práci s desetinnými čísly. Což byl hlavní nedostatek MMX. Také byl rozšířen počet registrů a jejich velikost a to o osm 128 bitových registrů. Tyto registry nejsou nijak sdílené, jako tomu bylo u MMX a slouží pouze pro účely instrukční sady SSE, každý dokáže pojmout čtyři 32 bitové proměnné typu float. V praxi se ovšem využívá datový typ double a to hlavně z důvodu absence funkcí pro typ float. Nezávislost nových registrů umožňuje libovolné přepínání mezi MMX a SSE dokonce lze využívat obě sady současně. Samozřejmostí je také libovolné přepínání z a do kódu x87. Tyto vlastnosti umožnily obrovský rozmach instrukcí SSE do multimédií a výrazné zrychlení jejich zpracování.

Vývoj pokračoval v roce 2001, kdy společnost Intel uvedla instrukční sadu SSE2. Tato sada rozšiřovala původní SSE o dalších 144 nových instrukcí. Prvním procesorem s touto instrukční sadou byl Pentium 4. Společnost AMD SSE2 uvedla až v procesorech Opteron v dubnu 2003. Instrukční sada SSE2 rozšiřuje flexibilitu registru a umožňuje využívat mnoho datových typů. SSE2 je již úplný systém umožňující masivní paralelizaci. Tuto instrukční sadu využijeme v kapitole 9 při implementaci našeho algoritmu.

Následovalo uvedení instrukční sady SSE3 ta je ovšem pouze malým vylepšením SSE2. Hlavním problémem teto sady a dalších jejich nástupkyň je obrovské množství uživatelů s procesory podporující maximálně SSE2. To se projevuje hlavně nabídkou softwaru, který využívá tyto novější instrukční sady. Aktuální verze nese označení SSE4.2 a obsahuje přes 700 instrukcí. Zájemce o kompletní přehled mohu odkázat na manuál společnosti Intel [25] ovšem kompletně se jím prokousat spolkne poměrně hodně času, zabírá totiž úctyhodných 4161 stran a je distribuován pouze v elektronické formě. Méně odvážné pak na některou z prací [26] nebo [27].

8.2.4 CUDA

CUDA je sada knihoven představená roku 2006 určená pro paralelizaci výpočtů za pomoci grafických karet od společnosti nvidia. Jak již bylo zmíněno výše grafické procesory mají oproti CPU velkou výhodu v počtu jader. To umožňuje dosahovat mnohem většího



(a) Porovnání GPU a CPU

(b) Jednotlivé vrstvy frameworku CUDA

Obrázek 9: CUDA

výpočetního výkonu v aplikacích, které využívají masivní paralelizaci. Standart CUDA je již poměrně rozšířený ovšem je určen jen pro GPU z dílny společnosti nvidia. To je důvod proč nemohl být využit v mé implementaci. CUDA sice může běžet v režimu simulované akcelerace ale k využití plného potenciálu je nutno vlastnit grafickou kartu osazenou čipem NVidia.

Nicméně i přes tyto omezení se jedná formu optimalizace, která stojí za úvahu. Pokud nám to hardware dovolí. Vzhledem k vyzrálosti této technologie je tak nesporným přínosem v oblasti výkonu stereo korespondenčních algoritmů a zpřístupňuje výpočetní výkon jednotky GPU na mnoha OS. Rozdíl oproti CPU je opravdu markantní, jak ukazuje graf na obrázku 9a. GPU má v dnešní době náskok v celém řadu a to GFLOPS v případě CPU a TFLOPS v případě GPU. Takovýto počet operací také vyžaduje dostatečný přísun dat. V tomto ohledu je aktuální stav srovnání CPU a GPU prakticky podobný jako v případě množství operací. V grafech se sice zpravidla uvádí teoretická hranice. Praktické výsledky ovšem potvrzují teoretické.

Z tohoto důvodu je poměrně škoda tento výkon nechat zahálet. Jeho využití nám umožni knihovny a různá API obsažené v CUDA. Nachází se zde také několik úrovní managementu, jak ukazuje obrázek 9b. Nejpohodlnější je využití střední úrovně nazvané Runtime. Vyšší úroveň obsahuje knihovny vyžité v CUDA. Nižší pak API k nastavení samotného ovladače a hardware.

Hlavním prvkem CUDA je rozšíření programovacího jazyka C za pomoci vlastního kompilátoru nvcc o definici kernelu a dalších prvků. Kernel zde reprezentuje jednotlivé paralelizované funkce. K dispozici jsou také vlákna a sdílená paměť tyto běžné techniky byly ovšem optimalizovány pro použití na GPU. Velikost režie vláken tudíž byla značně zredukována stejně jako penalizace přístupu do paměti. Samozřejmostí je také kompatibilita s OpenGL a Direct3D. Nvidia na svých stránkách [31] publikuje také poměrně obsáhle a dobře zpracované manuály.

8.2.5 OpenCL

OpenCL neboli je původně projekt společnosti Apple a řadí se po bok projektu CUDA prakticky se ovšem jedná o jeho nástupce. OpenCL obsahuje jednu podstatnou změnu ve filozofii. Má totiž za cíl umožnit provádět výpočty distribuovaně na heterogenních systémech skládajících se z více než jednoho procesoru. V praxi se zpravidla objevuje kombinace jednotek CPU a GPU. Tento projekt má také obrovskou výhodu a to ve formě podpory od mnoha výrobců mikročipů včetně Nvidie. Samotné principy fungovaní tohoto frameworku jsou velmi podobné jako ve frameworku CUDA.

Vzhledem k faktu že Apple postupně k vývoji přizval většinu velkých hráčů v oboru jako AMD, IBM, Intel a Nvidia. A nyní je tento framework spravován konsorciem Khronos [**?**] stejně jako třeba mnohem známější OpenGL. Navíc masivní podpora ze strany výrobců. Dělá z OpenCL jeden z nejlepších frameworků. Určených k vykonávání výpočtů na GPU. V blízké budoucnosti se bude pravděpodobně prosazovat čím dál tím více.

OpenCL implementuje standart C99 to znamená, že kód napsaný pro platformu OpenCL C je velmi podobný kódu v jazyce ANSI C. Jsou zde ovšem jisté změny například nemožnost využití rekurze. K překladu programů je popět nutno využit nestandardní překladač a to OpelCL Compiler ten je distribuován spolu s frameworkem OpenCL.

Na OpenCL je poznat zapojení firmy Nvidia. Základní jednotkou je zde také kernel a modely správy paměti i funkce jsou vesměs podobné. Ovšem při detailním studiu zjistíme že OpenCL poskytuje podstatně více možností. Zde bych opět doporučil manuály zveřejněné na stránkách projektu.

9 Implementace

K samotné implementaci algoritmu zvoleného algoritmu na bázi Belief propagation. Bylo využito jazyka C++ a o jak z důvodu multiplatformního využití tohoto jazyka ale také jeho přehlednosti a obrovským možnostem v oblasti optimalizace. Většina moderních technologií zabývajících se paralelními výpočty využívá přímo jazyk C nebo C++ popřípadě implementují normu ISO C99. Dalším rozhodujícím faktorem je dostupnost kvalitních knihoven pro práci s obrázky v této práci jsem zvolil již prověřenou knihovnu OpenCV.

Díky těmto knihovnám jsme se mohli soustředit na implementaci algoritmu BP a nezatěžovaly nás podružné problémy v podobě čtení dat ze souboru apod. Jistě se nabízí také využití jazyku jako je C# nebo JAVA. Ovšem vzhledem k běhu programu ve virtuálních strojích nedosahují tyto jazyky takového výkonu jako jazyky C a C++. Lépe řečeno virtuální stroj slouží spíše jako bezpečnostní prvek zajišťující oddělení operačního systému a programu. To má samozřejmě následek v podobě mírného nárůstu paměťové složitosti ale hlavně zpomalení vykonávání instrukcí programu. Jako příklad uveďme OS android se svým virtuálním strojem Dalvik. Program vykonávaný nativně bez tohoto virtuálního stroje v jazyce C byl při experimentech až čtyřikrát rychlejší než ten samý program jen napsaný v JAVA.

9.1 Implementace BP

Jak již bylo řečeno, algoritmus BP pracuje ve třech hlavních krocích.

1. Výpočet výchozích dat
2. Iterativní posílaní zpráv
3. Výpočet výsledku

V první fázi jsme implementovali čistý algoritmus BP bez jakékoliv optimalizace a jen se základními modifikacemi.

9.1.1 Výpočet výchozích dat

Výpočet výchozích dat probíhá poměrně jednoduchým způsobem, ten je popsán v kapitole 7.2. Zde si však řekneme něco o srovnávacích funkcích, které jsem využil k výpočtu vstupních dat. Také si povíme o reprezentaci MRF v mém programu.

Jednodušší funkce vyjádřená vzorcem 5 je pouhý součet absolutních rozdílů jednotlivých barevných kanálů.

$$C_{L,x_L}(d_x) = \sum_{c \in \{r,g,b\}} |L_c - R_c| \tag{5}$$

L_c reprezentuje hodnotu barevného kanálu pro daný bod v levém obrázku. Analogicky R_c označuje barevny kanál bodu z pravého obrázku. Výsledky získané za pomoci těchto

dat jsou uspokojivé. Odpovídají realitě a hlavně data získáme rychle bez složitých výpočtů.

Druhá funkce pak provádí výpočet o poznání složitější. Hlavním rozdílem oproti prvnímu vzorc 5 je zahrnutí okolních pixelů. Vzorec je vzhledem ke své složitosti definován po částech. První částí je obecná definice 6 rozdílu barevných kanálů Δ_{xy} dvou bodů x a y.

$$\Delta_{xy} = \left(\sum_{c \in \{r,g,b\}} |I_c(x) - I_c(y)| \right) \Big/ 3 \tag{6}$$

I_c v tomto vzorci definuje intenzitu barevného kanálů. Váha jednotlivých pixelů w_{xy} je pak definována jak barevným rozdílem, tak vzdáleností od centrálního pixelu. Popisuje ji vzorec 7.

$$w_{xy} = e^{-(\beta_{cw}^{-1}\Delta_{xy} + \gamma_{cw}^{-1}||x-y||_2)} \tag{7}$$

Hodnoty β_{cw} a γ_{cw} byly převzaty z práce [18] a definovány jsou na základě empirického zkoumání. A konečně kompletní funkce k ohodnocení shody dvou pixelů a jejich okolí 8.

$$C_{L,x_L}(d_x) = \frac{\sum_{(y_L,y_R)\in W_{x_L}\times W_{x_R}} w_{x_L,y_L} w_{x_R,y_R} d(y_L, y_R)}{\sum_{(y_L,y_R)\in W_{x_L}\times W_{x_R}} w_{x_L,y_L} w_{x_R,y_R}} \tag{8}$$

W_x reprezentuje pomocné okénko okolo pozorovaného pixelu a $d(y, y)$ Birchfieldovu a Tomasovu nepodobnost pixelu. Poté je na data aplikován uzávěr podle vzorce 9.

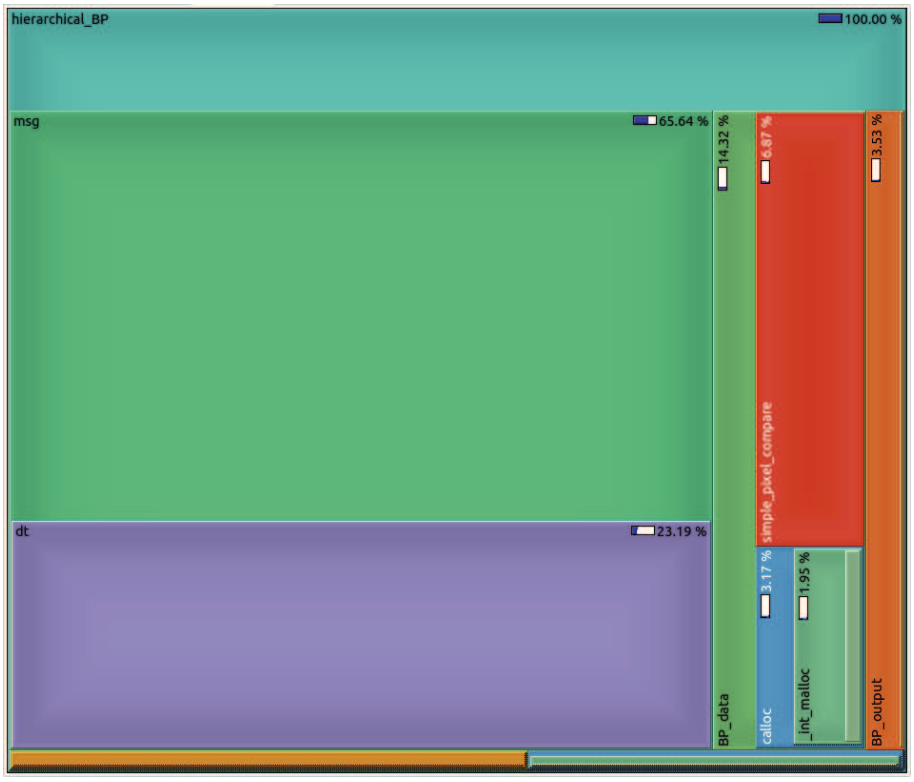
$$E^{(0)}_{D,x_l}(d_x) = \lambda_{bp} min(C_{L,x_l}(d_x), \eta_{bp}) \tag{9}$$

Tímto máme vytvořeny výchozí data. Při prvním pohledu na jednotlivé vzorce zistíme že je nutno pracovat s desetinými čísly proto jsou datat reprezentována datovým typem double nebo float. Volba datového typu zavísí na použité optimalizaci. Typ float je dostatečný avšak instrukce SSE pracují s typem double a při jejích využití je výhodnější využít tento datový typ.

Pro samotnou reprezentaci dat jsem zvolil jednoduché více rozměrné pole. Přesněji tří rozměrné pole při implementaci čistého BP a rychle konvergujícího BP. Pří implementaci hierarchického BP bylo nutno přidat další rozměr reprezentující jednotlivé úrovně komprese.

Představené způsoby výpočtu výchozích dat se diametrálně liší svou složitostí, jak ukazuje obrázek 10.

Na něm jsou mapy volání funkcí dvou implementací hierarchického algoritmu BP lišícího se pouze ve způsobu výpočtu výchozích dat. Jak můžeme vidět rozdíl doby výpočtu výchozích dat je opravdu značný. Naopak rozdíl ve výsledné disparitní mapě již tak znatelný není, viz příloha F. Obecně lze říci, že složitější funkce dosahuje spojitějších výsledku. Avšak doba výpočtu oproti jednoduchému rozdílu hodnot intenzity je až čtyřikrát delší. Proto je volba otázkou priorit dané implementace BP. Jedná-li se nám hlavně o čas zpracování určitě je vhodné využit pouze jednoduchý rozdíl intenzit.



(a) Graf volání pro jednoduchý rozdíl intenzit pixelu

(b) Graf volání pro složitější rozdíl pixelu

Obrázek 10: Grafy volání

9.1.2 Iterativní posílaní zpráv

V mé implementaci jsem využil zprávy typu sum product. Výpočet zprávy probíhá v několika fázích lze jej však souhrnně vyjádřit vzorcem 10.

$$m_{pq}^{t}(f_q) = \min_{f_p} \left(V(f_p, f_q) + D_p(f_p) + \sum_{s \in N(q)} m_{sp}^{t-1}(f_p) \right) \tag{10}$$

$V(f_p, f_q)$ označuje hodnotu hladkosti. $D_p(f_p)$ reprezentuju datovou složku zprávy. $\sum_{s \in N(q)} m_{sp}^{t-1}(f_p)$ odpovída součtu jednotlivých zpráv od sposedních pixelů. První se provede suma vstupních dat pro každou úroveň disparity. Vypočtené hodnoty se zapíší do cílových proměnných. Následně se na cílových proměnných provede normalizace a odstranění nevyhovujících hodnot vedoucích ke špatnému výsledku.

Požadavek inicializuje cíl, ten si od svého okolí vyžádá odpovídající zprávy. Počet iterací BP byl nastaven na 25. Což je dostatečná hodnota k dosažení správného výsledku. Ve fázi zasílaní zpráv lze aplikovat procedurální vylepšení popisované v kapitole 8.1.2 a tím vytvořit algoritmus spadající do třídy rychle konvergujících. V mém případě po fázi zasílaní zpráv z čerstvě vypočtených dat za pomoci vzorce 11 vypočtu hladinu důvěry.

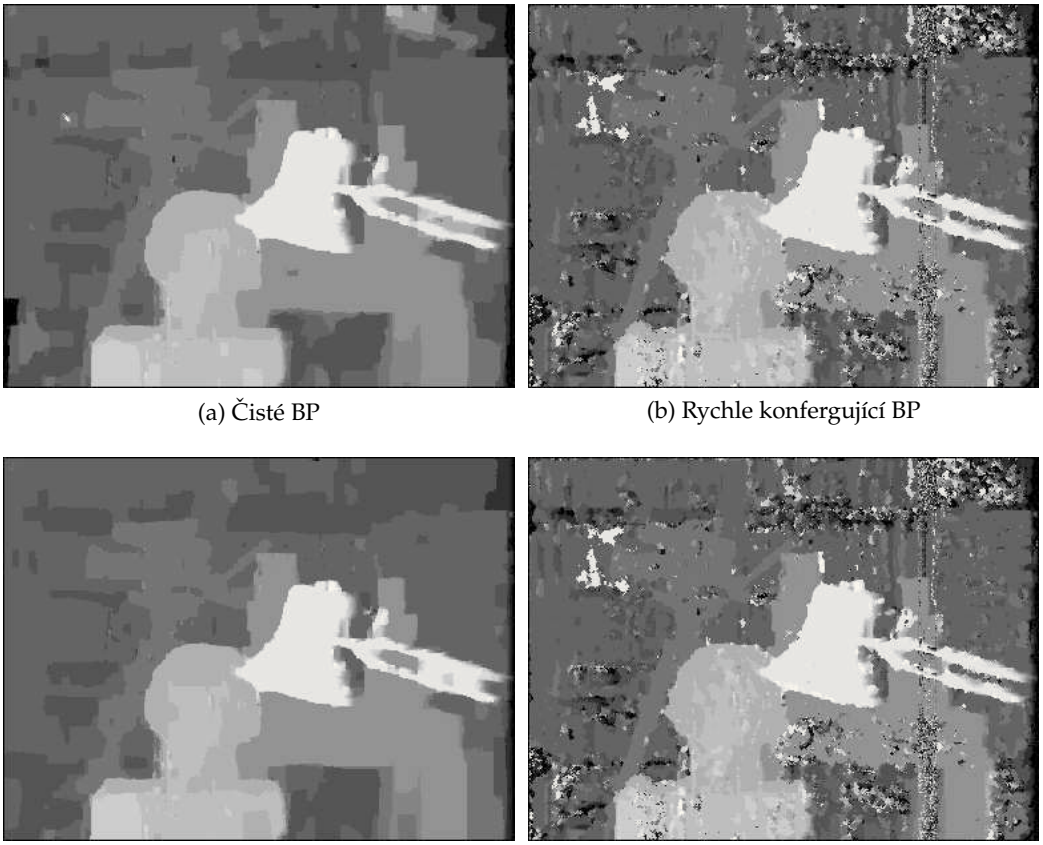
$$\left| \frac{C_L^1 - C_L^2}{C_L^2} \right| \tag{11}$$

C_L^1 reprezentuje nejnižší rozdíl daného vektoru neboli bodu. C_L^2 je pak druhý nejlepší. Výsledná hodnota reprezentuje velikost důvěry v aktuálně vypočtený výsledek pokud je tato hodnota větší než 0.4 je bod prohlášen za stabilní a další výpočty se pro něj neprovádějí. Hodnota byla určena experimentálně a převzata z práce [18]. Počet operací nutných k zaslání dostatečného množství zpráv můžeme snížit za pomoci implementace datové pyramidy popsané v kapitole 8.1.3. Jak je zmíněno v kapitole zabývající se datovou pyramidou zahrnutím tohoto vylepšení náš algoritmus získá přívlastek hierarchický.

Vlastní implementace datové pyramidy je prostá. Výchozí data se počítají zcela stejně. Pouze se umístí do nulové nekomprimované úrovně čtyř rozměrného pole. Vyšší úrovně se pak spočítají jednoduše podle principů popsaných v kapitole 8.1.3 jako funkci pro výpočet komprimované hodnoty jsem využil jednoduchou sumu rozdílu jednotlivých pixelů ve směru komprese 12. V tomto vzorci b reprezentoje množinu pixelů v korespondujícím kompresním okně. V případě dekomprese pak byla zvolena jednoduchá propagace dat.

$$D_b(f_b) = sum_{p \in b} D_p(f_b) \tag{12}$$

Tyto vylepšení jsou ovšem procedurálního charakteru. Jejich efekt na zkrácení doby zpracování je však značný a dosahované časy se ale pořád pouze blíží reálnému zpracování. Srovnání času běhu jednotlivých procedurálních modifikací lze získat z grafu na obrázku 23 v příloze E. Výsledné disparitní mapy jednotlivých optimalizací pak naleznete v přiloze F.



(a) Čisté BP (b) Rychle konfergující BP

(c) Hirearchické BP (d) Rychle konvergující hirearchické BP

Obrázek 11: Výsledné disparitní mapy jednotlivých implementací BP

9.1.3 Výpočet výsledku

Finální výsledek se vypočte po proběhnutí všech iterací nebo v případě rychle konvergujícího algoritmu po překročení určitého procenta stabilních pixelů v ideálním případě všech.

Výpočet vybírá pro každý bod nejlepší hodnotu disparity za pomoci sumy všech dostupných skrytých proměnných pro každou nezávislou proměnnou v MRF. Vybírá se nejnižší hodnota podle principu „vítěz bere vše“. Hodnotou disparity pak je hloubka ve které je nalezen vítěz.

Výsledná disparitní mapa se liší podle zvolené procesní implementace. Na obrázku 11 vidíme disparitní mapy jednotlivých algoritmů lišící se v implementaci procesu zasílaní zpráv. Všimněte si výrazných chyb vyskytujících se v implementaci rychle konvergujících algoritmů BP. Tyto chyby naznačují špatnou volbu hodnoty uzávěru nebo nevhoddnou formu výpočtu důvěry. I přes tyto chyby je výsledná disparitní mapa využi-

telná. nejlepších výsledků v poměru času a přesnosti však dosahuje hierarchická verze BP.

9.2 Paralelní zpracování

První částí optimalizace za pomocí paralelismu by vždy měla být analýza výchozího procedurálního problému. V našem případě vidíme na grafech volání z obrázků 10 převahu volání tří funkcí. Jedná se o jednotlivou zprávu msg, její uzávěr dt, a funkci srovnání dvou pixelů $pixel\_compare$. Vzhledem k velkému počtu těchto operací by určitě bylo vhodné, aby probíhaly souběžně. Dalším důležitým rozhodnutím je optimální volba paralelizovaného procedurálního řešení. Tato práce se snaží algoritmus BP optimalizovat hlavně po časové stránce. Z tohoto důvodu jsme zvolili pro výpočet dat pouze jednoduchý rozdíl intenzit pixelu. Dále jsem na základě grafu spotřeby paměti v čase vyobrazeném na obrázku 23 ten se nachází v příloze E zvolil k paralelizaci hierarchickou verzi BP. Především kvůli optimální době zpracování již v procedurální verzi, ale také kvůli kvalitním výsledným mapám disparity.

Nejprve jsem zavedl paralelismus na úrovni OS v podobě vláken. Ty jsme si představili v kapitole 8.2.2. Zde ovšem vyvstala otázka. V případě přílišného využívaní této metody totiž docházelo k zahlcení systému velkým množstvím vláken. Toto zahlcení měla na svědomí režie nutná ke správě obrovského počtu vláken. Tento problém jsem ovšem vyřešili pouze střídmým využitím vláken. Experimentálně jsem došel k paralelizaci za pomocí vláken na úrovních jednotlivých iterací BP a výpočtu výchozích dat. Použitím vláken na teto úrovni vzniknou v systému maximálně desítky vláken oproti stovkám, které vznikaly při využití tohoto způsobu pro každý řádek. Specificky v mé implementaci zpracovávám vždy najednou čtyři řádky dat. V případě výchozích dat je dodržena podmínka vzájemné nezávislosti jednotlivých srovnání. Vstupní data máme k dispozici všechna a jsou neměnná. Naopak v případě zpráv jsem využil stejný model ovšem za cenu mírné porušení nezávislosti. Správně by totiž výpočet zprávy měl probíhat na základě aktuálních dat vypočtených předešlými zprávami dané iterace. Já jsem zvolil model výpočtu založený na datech z předešlé iterace nikoli ze současné. Tato změna ovšem neměla na výsledek vliv. Uveďme si zde krátký výpis 1 na využití vláken v programovacím jazyce C.

```
//vypocet vychozich dat
for(i =0; i< BP_LEVELS;i++){
param[i] = i;
pthread_create(&vlakna[i], NULL, &bp_level_data, (void *) &param[i]);
}
//cekani na dokonceni vypoctu
for (i = 0; i < BP_LEVELS; i++) {
pthread_join(vlakna[i], NULL);
}
```

Výpis 1: Příklad vytváření vláken a následného vyčkávání na jejích dokončení.

Hlavní výhodou vláken je snadné sdílení paměti a bezproblémová komunikace mezi jednotlivými vlákny. Zde je důležité si správně rozvrhnout jednotlivé stavy vláken aby

nedocházelo k zbytečným systémovým voláním. Vlákna mají také svá omezení v podobě jisté fixace dané implementace na specifický OS. V mém případě se jedna o OS Linux a knihovnu pthread. Ta implementuje standart POSIX a je dostupná jak pro C tak i C++. Existuje i snadno dostupná verze této knihovny pro OS Windows tudíž napsaný kód lze využit na OS Linux tak na OS Windows.

Zavedení paralelismu na bázi vláken nezpůsobilo u menších vstupních dat výrazné snížení vypočetního času. V případě větších vstupů se už ovšem tato optimalizace projevila a to snížením doby zpracování přibližně o pět sekund. Množství uspořeného času velmi záleží na využitém hardwaru, OS tak i knihovně jenž byla k tomuto účelu využita.

Jako další úroveň paralelizace jsme uvažovali metodu SSE popsanou v kapitole 8.2.3. Využití teto technologie se nabízí ve fázi vypočtu třech nejfrekventovanějších funkcí z grafů volaní na obrázku 10. Vzhledem k možnosti spočítat všechny úrovně zprávy najednou. Za pomoci principu SIMD a navíc vstupní data těchto funkcí jsou již v požadovaném formátu dat. SSE umí pracovat pouze s jednorozměrnými poli. Dalším důvodem je že i malá úspora v této fázi bude mít velký efekt na celkový čas výpočtu a to vzhledem k obrovskému počtu volání teto funkce.

Kód programu v instrukcích SSE se piše přímo ve zdrojovém souboru jazyka C nebo C++ k jeho napojení slouží standartní knihovna. Samotný kód programu v SSE je podobný spíše jazyku Assembler jak ukazuje výpis 2. Oblast kódu v instrukcích SSE ohraničuje tag $\_asm$. Uvnitř tohoto tagu se využívají pouze SSE instrukce. Tag lze ovšem libovolně ukončit vykonat část kódu v jazyce C a opět pomocí tagu $\_asm$ přejít na instrukce SSE. $\_asm$ také neuvozuje pouze instrukce SSE ale také MMX a jiné sady procesosrových instrukci.

```
void *invertImageAsmSSE(unsigned char *src, unsigned char *dst, int imageSize){
  int numberOfLoops = imageSize/16;
  _asm{

    mov esi, src //esi = src
    mov edi, dst //edi = dst
    mov ecx, numberOfLoops //ecx obsahuje pocet opakovani
    pcmpeqb xmm2, xmm2 //do registru xmm2 da same jednicky

    loop1: //zacatek smycky
      movdqa xmm0, xmm2 //kopie samych nul do xmm0
      movdqu xmm1, [esi] //do xmm1 nacte 16 pixelu
      psubb xmm0, xmm1 //xmm0 = xmm0 - xmm1
      movdqu [edi], xmm0 //pixely ulozime do vysledneho obrazku
      add esi, 16 //posunuti v poli o 16
      add edi, 16 //posunuti v poli o 16
      add ecx, -1 //snizeni pocitadla smycek o 1
      jnz loop1 //pokud pocitadlo neni nulove, skoc na zacatek smycky
  }
}
```

Výpis 2: Příklad kód v instrukcích SSE. Inverze obrázku

Implemtaci této metody paralelizace jsem se zbýval pouze teoreticky.

Další zcela odlišnou variantou optimalizace je využití frameworku OpenCL ten už se následně postará o rozdělení a paralelizaci jednotlivých úkonů zpravidla za pomoci GPU. Tímto frameworkem se zběžně zabýváme v kapitole 8.2.5. Jak již bylo zmíněno tento framework využívá svůj vlatní jazyk implementující standart C99. Nutný je také speciální překladač opět dodávaný s frameworkem.

Hlavní jednotkou OpenCL je kernel ten v tomto kontextu reprezentuje paralelizovanou funkci provádějící daný výpočet. Ta je následně vykonána za pomoci kontextu a programu na výpočetní jednotce. Tento přístup může být zavadějící a to obzvláště pro začínajícího uživatele. Ovšem jakmile si zvykneme na jistou míru zobecnění neměl by tento přistup dělat problém. Tento v "chaos"je způsoben snahou o co největší možnou přenositelnost kódu mezi jednotlivými hardwarovými platformami. Jednoduchá ukázka kernelu se nachází ve výpisu 3

```
__kernel void vector_add_gpu (__global const float* src_a,
                              __global const float* src_b,
                              __global float* res,
                    const int num)
{
  const int idx = get_global_id (0);

  // Vykonavana operace
  if (idx < num)
    res[idx] = src_a[idx] + src_b[idx];
}
```

Výpis 3: Příklad jednoduch0ho kernelu OpenCL

K využití přikladu kernelu z výpisu 3 potřebujeme inicializovat celou řadu dalších komponent jejich představení a popis by vydalo na samostatnou publikaci jak ostatně ukazuje mimo jiné manuál zveřejněný na strámkach tohoto frameworku.

K implementaci touto metodou nedošlo z podobných důvodů jako v případě SSE.

10 Závěr

Závěrem práce můžeme konstatovat, že jsme splnili podmínky zadání. Tudíž jsme se seznámili s aktuálně používanými algoritmy a jejich funkčností. Také jsme navrhli vlastní algoritmus zpracovávající stereoskopický záznam scény v podobě dvou vstupních obrázků, jehož výstupem je disparitní mapa dané scény. Námi navržený algoritmus využívá postup známý jako Belief propagation a patří do kategorie hierarchických. Tento algoritmus byl jakožto stěžejní téma této práce popsána prozkoumán do detailů. Dále jsme provedli optimalizaci navrženého algoritmu za pomocí vhodně zvolených procesních vylepšení a správně umístěné paralelizace řešení problému.

Kdybychom ovšem měli pokračovat ve vylepšování našeho algoritmu. Určitě bychom mezi prvními vylepšeními měli uvažovat nad využitím segmentace obrazu na pomoci barevné informace. Této metody využívají také oba vzorové algoritmy [18, 19]. Další možností je využití uzávěrů důvěry popsaných v kapitole 8.1.2. Jako vhodná se také jeví implementace algoritmu Bleief propagation ve frameworku OpenCL.

Budoucnost tohoto algoritmu a jemu podobných je velmi nadějná možnosti jejich využití jsou prakticky omezeny pouze naší fantazií a dává lidstvu zcela nové možnosti v podobě značného rozšíření zraku a možnosti jeho simulace a to od poměrně malého měřítka v podobě robotiky a nanorobotiky až po to obrovské v podobě vesmírných dalekohledů a teleskopů na oběžné dráze.

Nové technologie procesoru otevřou ještě další možnosti v podobě skutečně realtime zpracování algoritmu. Ruku v ruce s tímto faktem půjde i miniaturizace zařízení, které bude schopno vykonat tyto algoritmy v dostatečně krátkém čase. V tomto ohledu bychom mohli zmínit novou technologii 3D tranzistorů ta umožní další miniaturizaci a nárůst počtu tranzistorů v mikročipech a zároveň se snížením spotřeby a množstvím odpadního tepla.

Dalším velkým přelomem může být uvedení procesoru s podporou masivní paralelizace. Společnost Intel nyní plánuje konstrukci procesorů s vice jak 50 jádry. Takovýto procesor by dosahoval až neuvěřitelně nízkých časů zpracování stereo korespondenčních algoritmů. V tomto případě by ovšem byly více zvýhodněný jiné způsoby řešení stereo korespondence. A v neposlední řadě je zde i dlouho diskutovaná technologie kvantových procesorů ta pravděpodobně otevře zcela nové brány oboru informatiky. A způsoby řešení našeho problémů jistě změní.

Ovšem nové technologie mohou také výrazně změnit požadavky uživatelů. Například uveďme technologii nového druhu fotoaparátu Lytro. Ten již nevyužívá klasický plochý mikročip zaznamenávající pouze intenzitu paprsku světla dopadajících na plochu čipu. Lytro obsahuje takzvaný Light field sensor ten nezaznamenává pouze intenzitu ale také vektor a jiné informace o paprsku světla který prochází tímto senzorem. To umožňuje zaznamenat snímek scény kompletně ve 3D i jedním přístrojem. Algoritmus rekonstruující tuto scénu bude pracovat na zcela odlišném principu. Snímek z tohoto typu fotoaparátu se neostří na určitý objekt fáze ostření probíhá až po fázi záznamu a za pomocí softwaru na PC. Tím by se náš algoritmus BP mohl dostat mino hlavní oblast zájmu. Ovšem z opačného úhlu pohledu stereoskopická soustava sestavená z tohoto

typu kamer by jistě poskytla nové možnosti a způsoby zpracování stereoskopických algoritmů.

11 Reference

[1] CYGANEK, Bogusław a J SIEBERT, *An introduction to 3D computer vision techniques and algorithms,* U.K.: J. Wiley, 2009. ISBN 04-700-1704-X

[2] KRUMNIKL, Michal, *Prostorové vidění ,* 2010 [cit. 2012-02-26]. Bakalářská práce. Univerzita Palackého, Filozofická fakulta. Vedoucí práce Alena Plháková. Dostupné z: http://theses.cz/id/psojl0/

[3] BROWN, M.Z., D. BURSCHKA a G.D. HAGER, *Advances in computational stereo ,* IEEE Transactions on Pattern Analysis and Machine Intelligence. 2003, DOI: 10.1109/TPAMI.2003.1217603.

[4] S.T. Barnard a M.A. Fischler, *Computational Stereo,* ACM Computing Surveys, vol. 14, pp. 553-572, 1982.

[5] S.T. U.R. Dhond a J.K. Aggarwal, *Structure from Stereo—A Review,* IEEE Trans. Systems, Man, and Cybernetics, vol. 19, pp. 1489-1510, 1989.

[6] A. Koschan, *What is New in Computational Stereo Since 1989: A Survey of Current Stereo Papers,* Technical Report 93-22, Technical Univ. of Berlin, 1993.

[7] Richard Hartley a Andrew Zisserman, *Multiple View Geometry in Computer Vision,* CUP, Cambridge, UK, 2003, DOI: 10.1017/S0263574705211621.

[8] FAUGERAS, Olivier, Quang-Tuan LUONG a Théo PAPADOPOULO, *The geometry of multiple images: the laws that govern the formation of multiple images of a scene and some of their applications,* Cambridge, Mass.: MIT Press, c2001, 644 s. ISBN 02-620-6220-8.

[9] D. Scharstein a R. Szeliski, *A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms,* Int'l J. Computer Vision, vol. 47, no. 1, pp. 7-42, 2002.

[10] T. Kanade and M. Okutomi, *A Stereo Matching Algorithm with an Adaptive Window: Theory and Experiment,* Proceedings of the 1991 IEEE International Conference on Robotics and Automation (ICRA '91), April, 1991, pp. 1088-1095.

[11] UNIVERSITY OF HAWAI'I, *Pan-STARRS,* [online]. 2005 [cit. 2012-04-22]. Dostupné z: http://pan-starrs.ifa.hawaii.edu/public/

[12] COMPAÑ, Patricia, Rosana SATORRE a Ramón RIZO, *Disparity estimation in stereoscopic vision by simulated annealing,*[online]. [cit. 2012-04-28]. DOI: 10.1108/03684920610662610. Dostupné z: http://rua.ua.es/dspace/bitstream/10045/795/1/ccia2003Ing.pdf

[13] Yuri BOYKOV, Olga VEKSLER a Ramin ZABIH, *Fast Approximate Energy Minimization via Graph Cuts,* IEEE Transactions on Pattern Analysis and Machine Intelligence archive Volume 23 Issue 11, November 2001

[14] LUKÁŠ, Václav, *Hledání korespondence v obrazech*,Ostrava, 2008. 200803038. Dostupné z: http://dspace.vsb.cz/handle/10084/68132. Diplomová práce. Vysoká škola báňská - Technická univerzita Ostrava. Vedoucí práce Eduard Sojka.

[15] HIRSCHMULLER, Heiko, *Accurate and efficient stereo processing by semi-global matching and mutual information*,2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Los Alamitos, Calif.: IEEE Computer Society, c2005, 807 - 814. ISBN 0-7695-2372-2.

[16] BYSTRZYCKI, Petr, *Stereo korespondence* [online]. 2010 [cit. 2012-04-26]. Bakalářská práce. Vysoká škola báňská - Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky. Vedoucí práce Michal Krumnikl. Dostupné z: ¡http://theses.cz/id/l3erav/¿.

[17] PEARL Judea, *Reverend Bayes on inference engines: A distributed hierarchical approach*, Proceedings of the American Association of Artificial Intelligence National Conference on AI Pittsburgh, PA: (1982) , p. 133–136.

[18] KLAUS, Andreas, Mario SORMANN, Konrad KARNER, *Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure*, IEEE transactions on pattern analysis and machine intelligence. 2008, č. 31. ISSN 0162-8828. Dostupné z: http://old.vrvis.at/2d3d/technology/stereomatching/images/segment-based-stereomatching.pdf

[19] YANG, Q., L. WANG, R. YANG, H. STEWÉNIUS a D. NISTÉR, *Stereo Matching with Color-Weighted Correlation, Hierarchical Belief Propagation, and Occlusion Handling*,IEEE transactions on pattern analysis and machine intelligence. 2009, č. 31. ISSN 0162-8828. Dostupné z: http://vision.ai.uiuc.edu/ qyang6/publications/pami-08-qingxiong-yang.pdf

[20] CZECH NATIONAL TEAM O.S., *Czech National Team*, [online]. 2006 [cit. 2012-04-24]. Dostupné z: http://www.czechnationalteam.cz/

[21] TANENBAUM, Andrew S a Albert S WOODHULL, *Operating systems: design and implementation. 3rd ed.*, Upper Saddle River: Prentice Hall, 2006, 1054 s. ISBN 01-314-2938-8.

[22] D. Scharstein and R. Szeliski, *High-accuracy stereo depth maps using structured light*,IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2003), volume 1, pages 195-202, Madison, WI, June 2003.

[23] D. Scharstein and C. Pal, *Learning conditional random fields for stereo*,IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007), Minneapolis, MN, June 2007.

[24] H. Hirschmüller and D. Scharstein, *Evaluation of cost functions for stereo matching*,IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007), Minneapolis, MN, June 2007.

[25] INTEL, *Intel® 64 and IA-32 Architectures Software Developer's Manual*, [online]. Combined Volumes: 1, 2A, 2B, 2C, 3A, 3B and 3C. 2011 [cit. 2012-02-26]. Dostupné z: http://www.intel.com/content/www/us/en/architecture-and-technology/64-ia-32-architectures-software-developer-manual-325462.html.

[26] ŠURKALA, Milan, *Sada funkcí pro zpracování obrazu využívající instrukcí MMX a SSE*, [online]. Ostrava, 2007 [cit. 2012-02-26]. 200702439. Dostupné z: http://hdl.handle.net/10084/62217. Bakalářská práce. VŠB - TUO. FEI. Vedoucí práce Sojka Eduard.

[27] LOMIČ, Petr, *Knihovna funkcí pro zpracování obrazu využívající jednotky SSE*, [online]. Ostrava, 2008 [cit. 2012-02-26]. 200803234. Dostupné z: http://hdl.handle.net/10084/68328. Bakalářská práce. VŠB - TUO. FEI. Vedoucí práce Sojka Eduard.

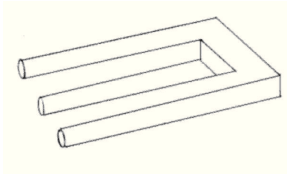
[28] *The Middlebury Computer Vision Pages*, [online]. [cit. 2012-03-16]. Dostupné z: http://vision.middlebury.edu/

[29] *Middlebury Stereo Datasets Pages*, [online]. [cit. 2012-03-16]. Dostupné z: http://vision.middlebury.edu/stereo/data/

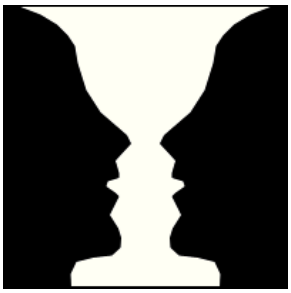
[30] *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, proceedings : 18-20 June, 2003, Madison, Wisconsin [online]. Los Alamitos, Calif.: IEEE Computer Society, 2003, s. 195-202 [cit. 2012-03-16]. ISBN 0769519008. Dostupné z: http://community.middlebury.edu/ schar/papers/structlight/

[31] NVIDIA, *NVIDIA GPU Computing Documentation: NVIDIA Developer Zone*, [online]. 2012 [cit. 2012-03-19]. Dostupné z: http://developer.nvidia.com/nvidia-gpu-computing-documentation

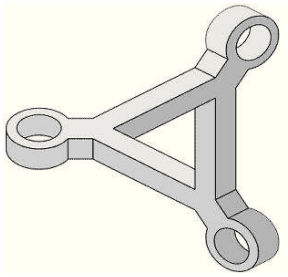
A Optické klamy



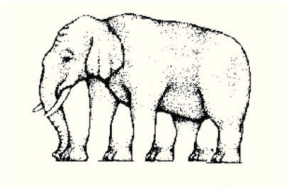
(a) Nemožný tvar



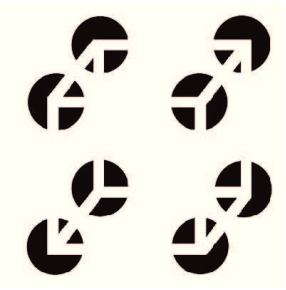
(b) Váza nebo tváře



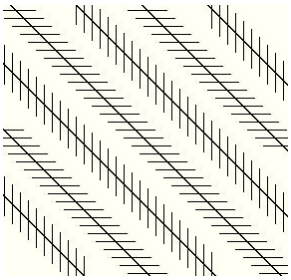
(c) Nemožný tvar



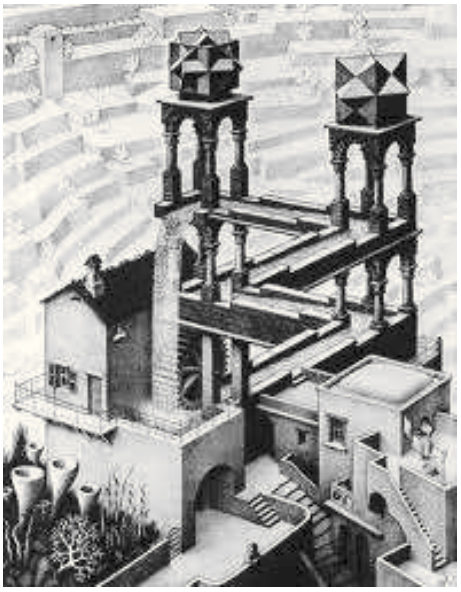
(d) Slon s více nohama



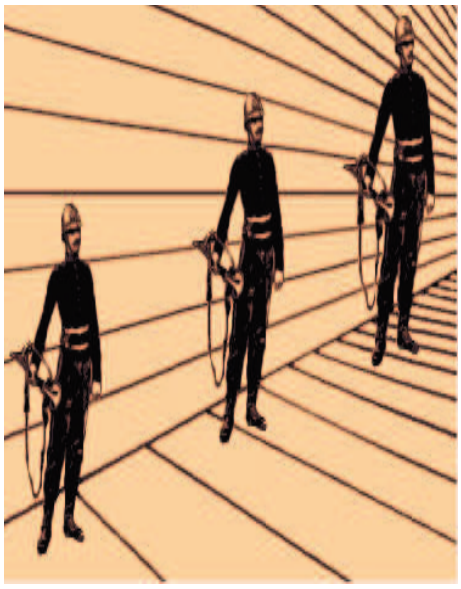
(e) Příklad schopnosti dokreslit objekty



(f) Iluze deformace rovnoběžných čar

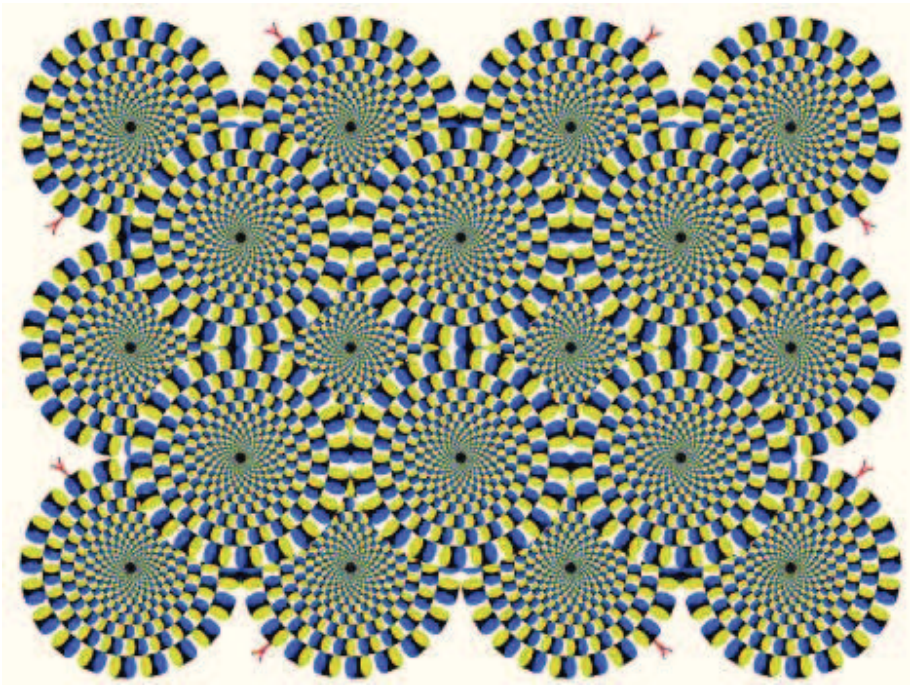


(g) Vodopády (MC Escher)



(h) Iluze změny velikosti objektu

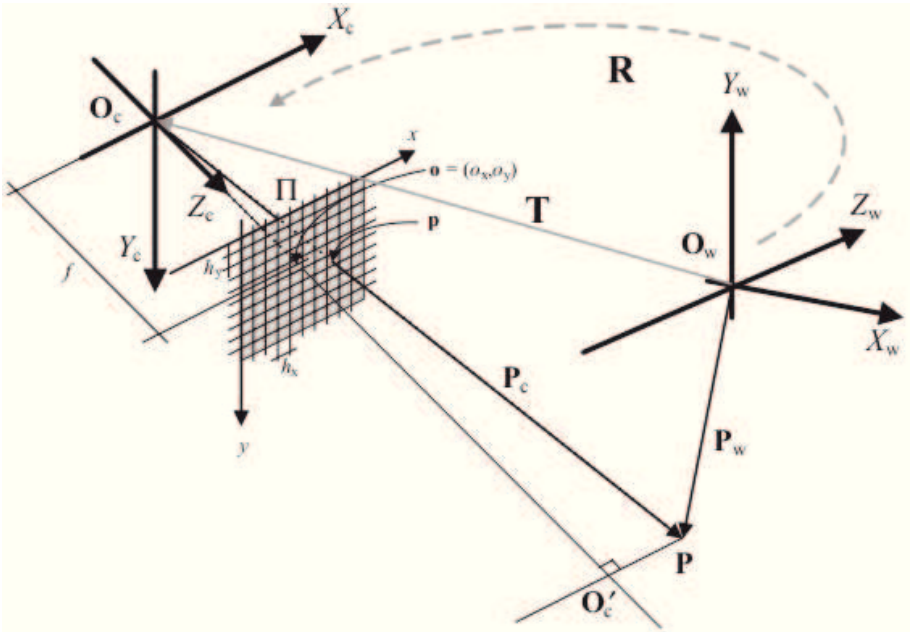
Obrázek 12: Optické klamy



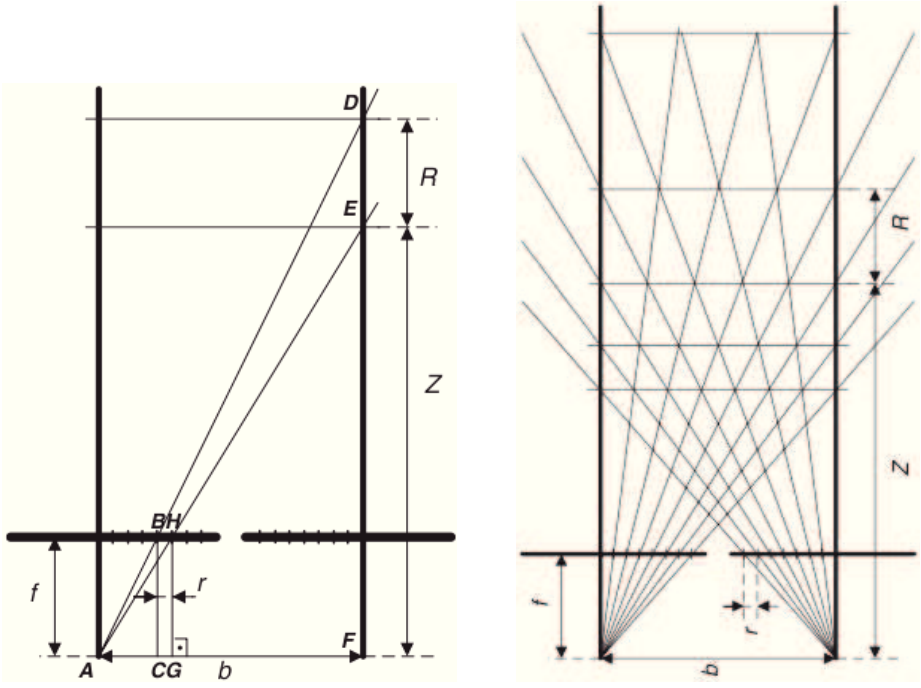
Obrázek 13: Iluze pohybu

Obrázek 14: Příklad stereogramu

B Epipolarni geometrie

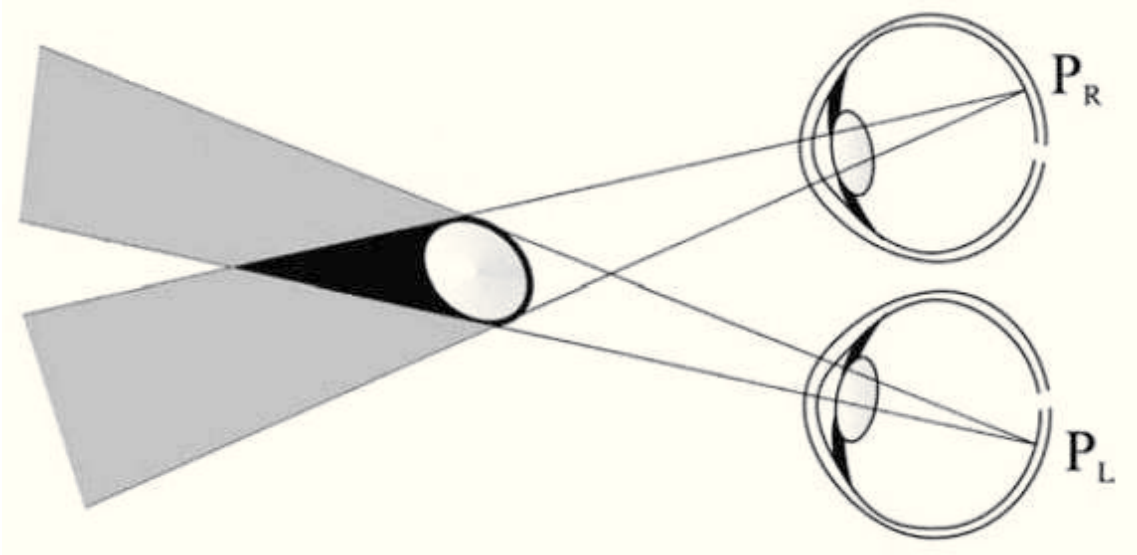


Obrázek 15: Pinhole model [1]

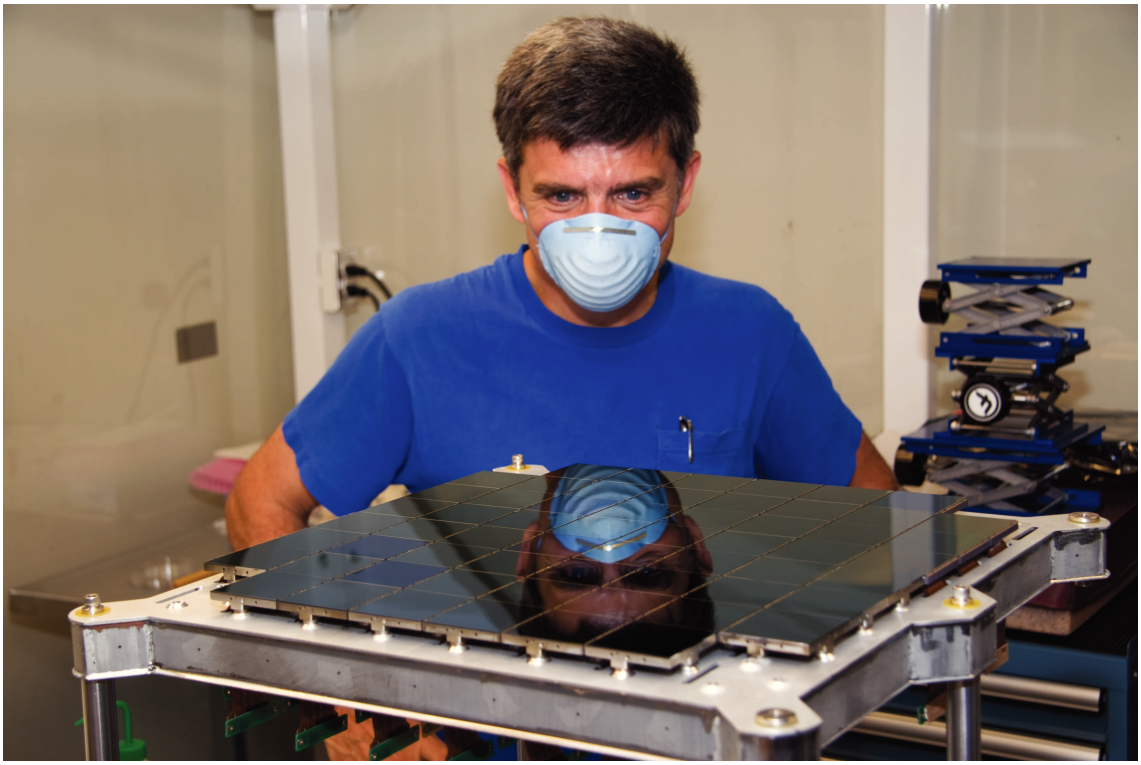


(a) (b)

Obrázek 16: Závislost chyby výpočtu na rozlišení [1]



Obrázek 17: Problém překrytí [1]



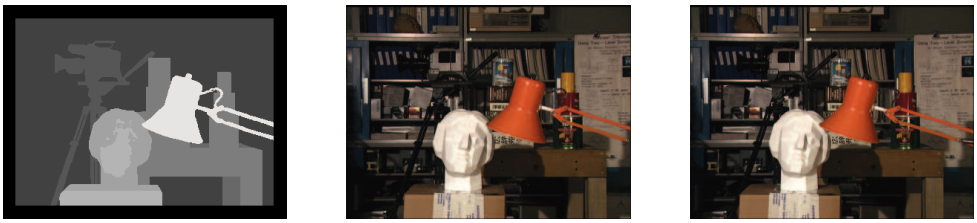
Obrázek 18: Největší CCD senzor

C Nejběžnější funkce k ohodnocení shody

D_{SAD} - suma absolutních rozdílů $D_{SAD} = \sum_{(i,j)\in U} \lvert I_l(x+i, y+j) - I_r(x+d_x+i, y+d_y+j)\rvert$
D_{ZSAD} - nula znamená SAD $D_{ZSAD} = \sum_{(i,j)\in U} \lvert(I_l(x+i, y+j) - \overline{I_l(x,y)}) - (I_r(x+d_x+i, y+d_y+j) - \overline{I_l(x+d_x, y9+d_y)})\rvert$
D_{SSD} - suma mocněných rozdílů $D_{SSD} = \sum_{(i,j)\in U}[I_l(x+i, y+j) - I_r(x+d_x+i, y+d_y+j)]^2$
D_{ZSSD} - nula znamená SSD $D_{ZSSD} = \sum_{(i,j)\in U}([I_l(x+i, y+j) - \overline{I_l(x,y)}] - [I_r(x+d_x+i, y+d_y+j) - \overline{I_l(x+d_x, y9+d_y)}])^2$
D_{SSD-N} - normované SSD $D_{SSD-N} = \frac{\sum_{(i,j)\in U}[I_l(x+i,y+j)-I_r(x+d_x+i,y+d_y+j)]^2}{\sqrt{\sum_{(i,j)\in U} I_l(x+i,y+j)^2 * \sum_{(i,j)\in U} I_r(x+d_x+i,y+d_y+j)^2}}$
D_{ZSSD-N} - nula znamená normované SSD $D_{ZSSD} = \frac{\sum_{(i,j)\in U}([I_l(x+i,y+j)-\overline{I_l(x,y)}]-[I_r(x+d_x+i,y+d_y+j)-\overline{I_l(x+d_x,y9+d_y)}])^2}{\sqrt{\sum_{(i,j)\in U}(I_l(x+i,y+j)-\overline{I_l(x,y)})^2 * \sum_{(i,j)\in U}([I_r(x+d_x+i,y+d_y+j)-\overline{I_r(x+d_x,y9+d_y)})^2}}$
D_{CV} - kovariace - variace $D_{CV} = \frac{\sum_{(i,j)\in U}(I_l(x+i,y+j)-\overline{I_l(x,y)}]-[I_r(x+d_x+i,y+d_y+j)-\overline{I_l(x+d_x,y9+d_y)})}{\sqrt{\sum_{(i,j)\in U}(I_l(x+i,y+j)-\overline{I_l(x,y)})^2 * \sum_{(i,j)\in U}([I_r(x+d_x+i,y+d_y+j)-\overline{I_r(x+d_x,y9+d_y)})^2}}$
D_{SCP} - suma křížových produktů $D_{SCP} = \sum_{(i,j)\in U} I_l(x+i, y+j) * I_r(x+d_x+i, y+d_y+j)$
D_{SCP-N} normované SCP $D_{SCP} = \frac{\sum_{(i,j)\in U} I_l(x+i,y+j) * I_r(x+d_x+i,y+d_y+j)}{\sqrt{\sum_{(i,j)\in U} I_l(x+i,y+j)^2 * \sum_{(i,j)\in U} I_r(x+d_x+i,y+d_y+j)^2}}$

Tabulka 2: Nejběžnější funkce k ohodnocení shody.

D Soubory dat



(a) Trueground (b) Levý pohled (c) Pravý pohled

Obrázek 19: Datový set Tsukuba [9]

(a) Trueground (b) Levý pohled (c) Pravý pohled

Obrázek 20: Datový set Cones [22]

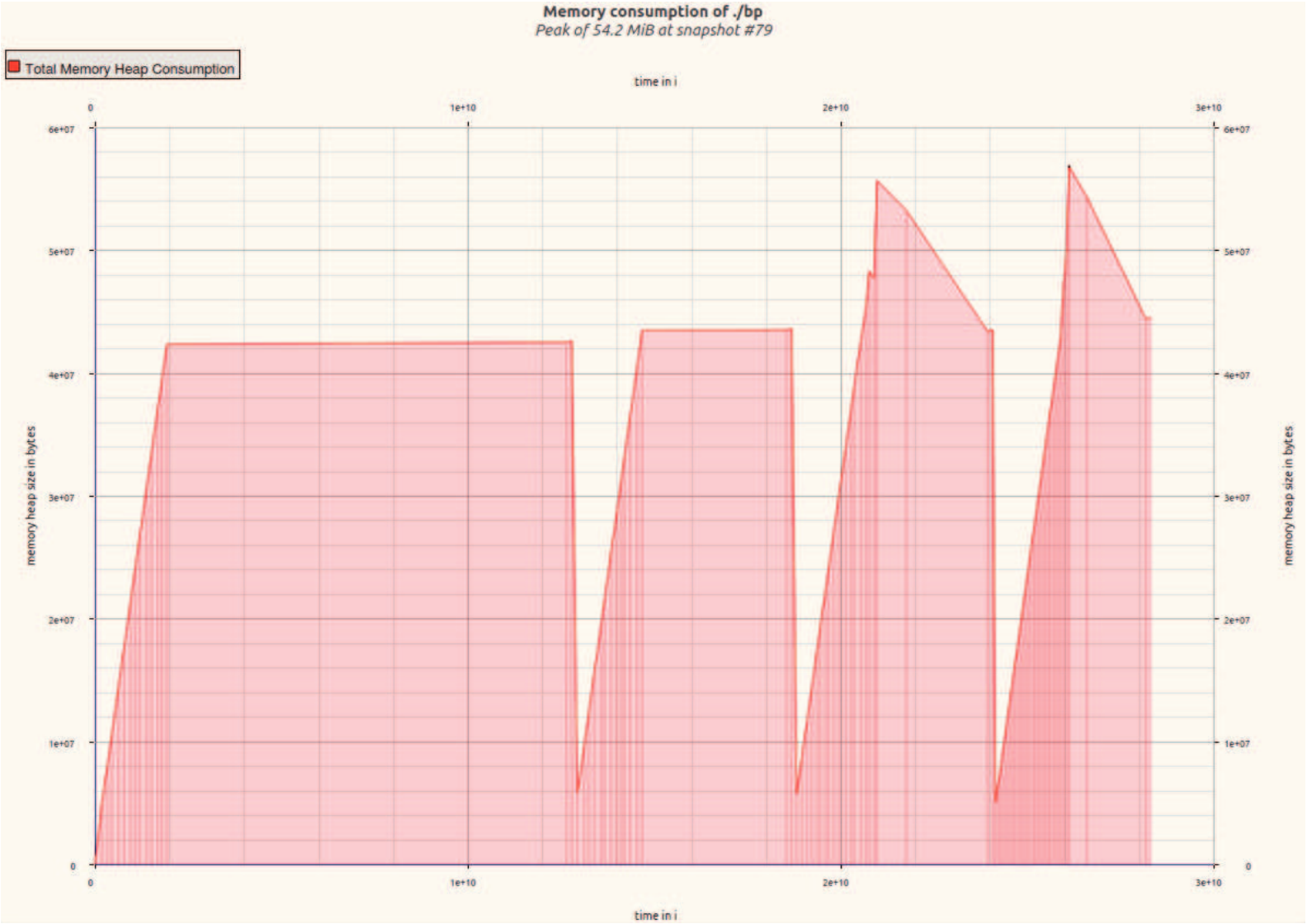
(a) Trueground (b) Levý pohled (c) Pravý pohled

Obrázek 21: Datový set Teddy [22]

(a) Trueground (b) Levý pohled (c) Pravý pohled

Obrázek 22: Datový set Venus [9]

E Graf využití paměti v čase

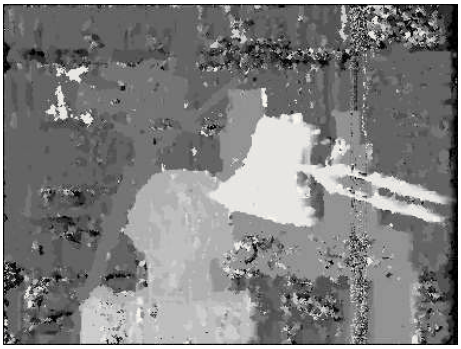


Obrázek 23: Graf využití paměti v čase. Z leva do prava jednotlivé části reprezentují čiste BP, rychle konvergující BP, hierarchické BP a rychle konvergující BP

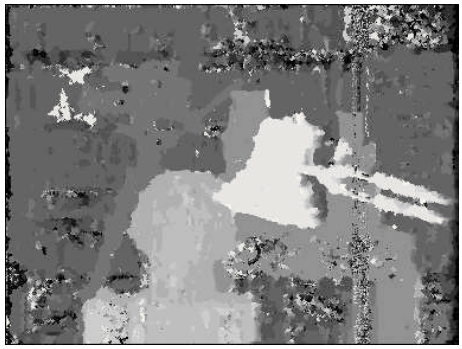
F Výsledné disparitní mapy

(a) Čisté BP s jednoduchým rozdílem intenzit

(b) Čisté BP se složitější funkcí shody



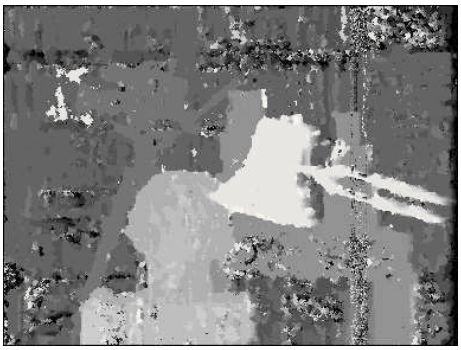
(c) Rychle konvergující BP s jednoduchým rozdílem intenzit



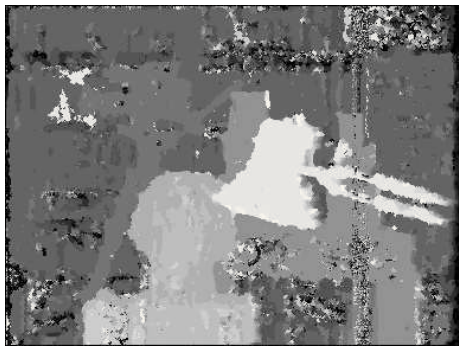
(d) Rychle konvergující BP se složitější funkcí shody

(e) Hierarchické BP s jednoduchým rozdílem intenzit

(f) Hierarchické BP se složitější funkcí shody



(g) Rychle konvergující hierarchické BP s jednoduchým rozdílem intenzit



(h) Rychle konvergující hierarchické BP se složitější funkcí shody

Obrázek 24: Výsledné disparitní mapy jednotlivých typů implementací BP pro datový set Tsukuba